(2)

SEC

AD-A202 963

RT DOCUMENTATION PAGE

DTIC FILE COPY

1a. UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88-1282
6a. NAME OF PERFORMING ORGANIZATION: Stanford University 6b. OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NE
6c. ADDRESS (City, State and ZIP Code): 660 Arguello Way Encina Hall Encina Hall Stanford, CA 94305-6060	7b. ADDRESS (City, State and ZIP Code): Bldg 410 Bolling AFB DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR/NE 8b. OFFICE SYMBOL (If applicable): NE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-88-0024
8c. ADDRESS (City, State and ZIP Code): Bldg 410 Bolling AFB DC 20332-6448	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 61102F PROJECT NO.: 2305 TASK NO.: B1 WORK UNIT NO.:
11. TITLE (Include Security Classification): Optical Computing Research	
12. PERSONAL AUTHOR(S): Professor Goodman	
13a. TYPE OF REPORT: Annual 13b. TIME COVERED: FROM 01Oct87 TO 30Sept 14. DATE OF REPORT (Yr., Mo., Day): 88 15. PAGE COUNT	
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES: FIELD / GROUP / SUB. GR. 18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The research focused on understanding the global as well as local properties of the neural network model. Global properties are the dynamics of the network, convergency properties, computational power and capacity. By local properties we mean the theory of threshold logic elements, the basic building blocks of the network.

Stapl

DTIC ELECTE DEC 16 1988 S D

UNCLASSIFIED

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: GILES	22b. TELEPHONE NUMBER (Include Area Code): (202) 767-4931 22c. OFFICE SYMBOL: NE

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

88 12 15 003

SECURITY CLASSIFICATION OF THIS PAGE

ANNUAL REPORT

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

CONTRACT # AFOSR-88-0024

Joseph W. Goodman
Jehosua Bruck

October, 1988

The work performed on this grant during the first grant year has fallen into two categories:

1. Theoretical studies of neural networks; and

2. Fundamental studies of optical interconnects.

We report on the progress in both of these areas in the the two following sections.

Neural Networks

The research this year focused on understanding the global as well as local properties of the neural network model. Global properties are the dynamics of the network, convergence properties, computational power and capacity. By local properties we mean the theory of threshold logic elements, the basic building blocks of the network. Here we mention only two main contributions this year. The first relates to the global properties while the second to the local properties of the neural network model. The details of these contributions appear in [1] and [2].

In [1] we investigated the relation between error-correcting codes and neural networks. The motivation behind this work was that a neural network model can be viewed as a decoder. The stable states correspond to codewords, the probe vector corresponds to the received vector, and convergence to the closest stable state corresponds to Maximum Likelihood Decoding (MLD). We found several natural

ways of connecting the concept of error correcting codes with the concept of neural networks. In particular, we showed that the MLD problem in a linear block code is equivalent to finding the global maximum of the energy function of a neural network that can be easily constructed knowing the basis set of the code. We also have a dual result: given a linear block code, we can easily construct a neural network in which every local energy maximum corresponds to a codeword and every codeword corresponds to a local maximum, thus solving the 'programming' problem for linear codes. The results are generalized for both nonbinary and nonlinear codes.

In [2] we answered a fundamental question in the theory of threshold logic. Suppose that instead of using a linear threshold (LT) element we use a polynomial threshold (PT) element. A PT element computes a polynomial (instead of a linear form) with the restriction that the number of terms in the polynomial is 'small' (the number of terms is bounded by some polynomial in the number of variables). The question is: what is the power of a PT element and how does it compare with that of a LT element? The answer is that we do not gain much by using PT's instead of LT's. A 'small' two layer network of LT's can do strictly more that a single PT element can do. In order to answer this question we developed a novel technique based on harmonic analysis and derived bounds on the number of terms in the PT representation. As a byproduct we also found a new way of deriving counting results.

The above results are important to both the theory of neural networks and the area of circuit complexity in the theory of computer science.

References

[1] J. Bruck and M. Blaum, "Neural Networks, Error-Correcting Codes and Polynomials over the Binary n-Cube", accepted for publication in the *IEEE Transactions on Information Theory.*

[2] J. Bruck, "Harmonic Analysis of Polynomial Threshold Functions", submitted to *SIAM Journal on Discrete Mathematics.*

Optical Interconnections

Our work on fundamental properties of optical interconnections contained two different subtasks. First, we have nearly competed a study of the comparison of coaxial cable

with optical interconnects for use within multiprocessor machines. The analysis is a rigorous one, and considers many different cases regarding termination, source impedance, etc. When complete, it will provide guidelines for when it is appropriate to use optics for this type of interconnection and when it may not be.

A second task has been the examination of a recent paper by S.H. Lee and his group in which projections of the advantages of optics at the intra-chip level of interconnects were made. These projections were far more optimistic than our own previous projections, and it was important to discover the difference between the two analyses and the reasons for their different predictions. One difference lies in the fact that our considerations were fundamental ones while theirs were more practical. However, this difference did not explain the difference of the predictions. We discovered that, while our own predictions had projected the capabilities of both optics and electronics to the future, the San Diego group had projected only the capabilities of optics to the future. Characteristics assumed of electronics were not similarly projected (for example, 3 μm technology was assumed).

Future Activities

Our activities in the coming year will consist of the following :

1. Continuation of our fundamental theoretical studies of neural networks;
2. A beginning of a study of interconnect limits in GaAs technology (all of our previous studies have been for silicon).

Publications and Presentations at Meetings

1. J. Bruck and J. Sanz, "A study on Neural Networks", *International Journal of Intelligent Systems*, Vol. 3, pp. 59075 (1988)

2. J. Bruck and J.W. Goodman, "A Generalized Convergence Theorem for Neural Networks and its Applications in Combinatorial Optimization", *Proc. First IEEE International Conference on Neural Networks*, San Diego, pp. III-649-656, (1987).

3. J. Bruck and J.W. Goodman, "A Generalized Convergence Theorem for Neural Networks", accepted for publication in *IEEE Trans. on Information Theory*.

4. J. Bruck and J.W. Goodman, "On the Power of Neural Networks for Solving Hard problems", *IEEE Conference on Neural Information Processing Systems,* Denver, Colorado, November 1987. Submitted to the *Journal of Complexity.*

5. J. Bruck and M. Blaum, "Neural Networks, Error-Correcting Codes and Polynomials over the Binary n-Cube", accepted for publication in the *IEEE Transactions on Information Theory.* Also presented at ISIT in Japan.

6. J. Bruck, "Harmonic Analysis of Polynomial Threshold Functions", submitted to *SIAM Journal on Discrete Mathematics.*

7. J. W. Goodman, "Optics as an Interconnect Technology", to appear in *Optical Computing and Optical Information Processing,* H.H. Arsenault, Editor, Academic Press.

On the Power of Neural Networks for Solving Hard Problems *

Jehoshua Bruck
Joseph W. Goodman
Information Systems Laboratory
Department of Electrical Engineering
Stanford University
Stanford, CA 94305

Abstract

This paper deals with a neural network model in which each neuron performs a threshold logic function. An important property of the model is that it always converges to a stable state when operating in a serial mode [2,5]. This property is the basis of the potential applications of the model such as associative memory devices and combinatorial optimization [3,6].

One of the motivations for use of the model for solving hard combinatorial problems is the fact that it can be implemented by optical devices and thus operate at a higher speed than conventional electronics.

The main theme in this work is to investigate the power of the model for solving NP-hard problems [4,8], and to understand the relation between speed of operation and the size of a neural network. In particular, it will be shown that for any NP-hard problem the existence of a polynomial size network that solves it implies that NP=co-NP. Also, for Traveling Salesman Problem (TSP), even a polynomial size network that gets an ϵ-approximate solution does not exist unless P=NP.

The above results are of great practical interest, because right now it is possible to build neural networks which will operate fast but are limited in the number of neurons.

*Presented at the IEEE Neural Information Processing Systems Conference, Denver, Colorado, November 1987.

1 Background

The neural network model is a discrete time system that can be represented by a weighted and undirected graph. There is a weight attached to each edge of the graph and a threshold value attached to each node (neuron) of the graph. The *order* of the network is the number of nodes in the corresponding graph. Let N be a neural network of order n; then N is uniquely defined by (W,T) where:

- W is an $n \times n$ symmetric matrix, W_{ij} is equal to the weight attached to edge (i,j).
- T is a vector of dimension n, T_i denotes the threshold attached to node i.

Every node (neuron) can be in one of two possible states, either 1 or -1. The state of node i at time t is denoted by $V_i(t)$. The *state* of the neural network at time t is the vector $V(t)$.

The next state of a node is computed by:

$$V_i(t+1) = sgn(H_i(t)) = \begin{cases} 1 & \text{if } H_i(t) \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

where

$$H_i(t) = \sum_{j=1}^{n} W_{ji} V_j(t) - T_i$$

The next state of the network, i.e. $V(t+1)$, is computed from the current state by performing the evaluation (1) at a subset of the nodes of the network, to be denoted by S. The modes of operation are determined by the method by which the set S is selected in each time interval. If the computation is performed at a single node in any time interval, i.e. $\mid S \mid = 1$, then we will say that the network is operating in a *serial* mode; if $\mid S \mid = n$ then we will say that that the network is operating in a *fully parallel* mode. All the other cases, i.e. $1 <\mid S \mid < n$ will be called *parallel* modes of operation. The set S can be chosen at random or according to some deterministic rule.

A state $V(t)$ is called *stable* iff $V(t) = sgn(WV(t) - T)$, i.e. there is no change in the state of the network no matter what the mode of operation is. One of the most important properties of the model is the fact that it always converges to a stable state while operating in a serial mode. The main idea in the proof of the convergence property is to define a so called *energy function* and to show that this energy function is nondecreasing when the state of the network changes. The energy function is:

$$E(t) = V^T(t)WV(t) - 2V^T(t)T \tag{2}$$

An important note is that originally the energy function was defined such that it is nonincreasing [5]; we changed it such that it will comply with some known graph problems (e.g. Min Cut).

A neural network will always get to a stable state which corresponds to a local maximum in the energy function. This suggests the use of the network as a device for performing a local search algorithm for finding a maximal value of the energy function [6]. Thus, the network will perform a local search by operating in a random and serial mode. It is also known [2,9] that maximization of E associated with a given network N in which $T = 0$ is equivalent to finding the Minimum Cut in N. Actually, many hard problems can be formulated as maximization of a quadratic form (e.g. TSP [6]) and thus can be mapped to a neural network.

2 The Main Results

The set of stable states is the set of possible final solutions that one will get using the above approach. These final solutions correspond to local maxima of the energy function but do not necessarily correspond to global optima of the corresponding problem. The main question is: suppose we allow the network to operate for a very long time until it converges; can we do better than just getting some local optimum? i.e., is it possible to design a network which will always find the exact solution (or some guaranteed approximation) of the problem?

Definition: Let X be an instance of problem. Then $\mid X \mid$ denotes the size of X, that is, the number of bits required to represent X. For example, for X being an instance of TSP, $\mid X \mid$ is the number of bits needed to represent the matrix of the distances between cities.

Definition: Let N be a neural network. Then $\mid N \mid$ denotes the size of the network N. Namely, the number of bits needed to represent W and T.

Let us start by defining the desired setup for using the neural network as a model for solving hard problems.

Consider an optimization problem L, we would like to have for every instance X of L a neural network N_X with the following properties:

- Every local maximum of the energy function associated with N_X corresponds to a global optimum of X.
- The network N_X is small, that is, $\mid N_X \mid$ is bounded by some polynomial in $\mid X \mid$.

Moreover, we would like to have an algorithm, to be denoted by A_L, which given an instance $X \in L$, generates the description for N_X in polynomial (in $\mid X \mid$)

time.

Now, we will define the desired setup for using the neural network as a model for finding approximate solutions for hard problems.

Definition: Let E_{glo} be the global maximum of the energy function. Let E_{loc} be a local maximum of the energy function. We will say that a local maximum is an ϵ-approximate of the global iff:

$$\frac{E_{glo} - E_{loc}}{E_{glo}} \leq \epsilon$$

The setup for finding approximate solutions is similar to the one for finding exact solutions. For $\epsilon \geq 0$ being some fixed number. We would like to have a network N_{X_ϵ} in which every local maximum is an ϵ-approximate of the global and that the global corresponds to an optimum of X. The network N_{X_ϵ} should be small, namely, $\mid N_{X_\epsilon} \mid$ should be bounded by a polynomial in $\mid X \mid$. Also, we would like to have an algorithm A_{L_ϵ}, such that, given an instance $X \in L$, it generates the description for N_{X_ϵ} in polynomial (in $\mid X \mid$) time.

Note that in both the exact case and the approximate case we do not put any restriction on the time it takes the network to converge to a solution (it can be exponential).

At this point the reader should convince himself that the above description is what he imagined as the setup for using the neural network model for solving hard problems, because that is what the following definition is about.

Definition: We will say that a neural network for solving (or finding an ϵ-approximation of) a problem L exists if the algorithm A_L (or A_{L_ϵ}) which generates the description of N_X (or N_{X_ϵ}) exists.

The main results in the paper are summarized by the following two propositions. The first one deals with exact solutions of NP-hard problems while the second deals with approximate solutions to TSP.

Proposition 1 *Let L be an NP-hard problem. Then the existence of a neural network for solving L implies that NP = co-NP.*

Proposition 2 *Let $\epsilon \geq 0$ be some fixed number. The existence of a neural network for finding an ϵ-approximate solution to TSP implies that P=NP.*

Both (P=NP) and (NP=co-NP) are believed to be false statements, hence, we can not use the model in the way we imagine.

The key observation for proving the above propositions is the fact that a single iteration in a neural network takes time which is bounded by a polynomial in the size of the instance of the corresponding problem. The proofs of the above two propositions follow directly from known results in complexity theory and should not be considered as new results in complexity theory.

3 The Proofs

Proof of Proposition 1: The proof follows from the definition of the classes NP and co-NP, and Lemma 1. The definitions and the lemma appear in Chapters 15 and 16 in [8] and also in Chapters 2 and 7 in [4].

Lemma 1 *If the complement of an NP-complete problem is in NP, then NP=co-NP.*

Let L be an NP-hard problem. Suppose there exists a neural network that solves L. Let $\hat{L}$ be an NP-complete problem. By definition, $\hat{L}$ can be polynomialy reduced to L. Thus, for every instance $X \in \hat{L}$, we have a neural network such that from any of its global maxima we can efficiently recognize whether X is a 'yes' or a 'no' instance of $\hat{L}$.
We claim that we have a nondeterministic polynomial time algorithm to decide that a given instance $X \in \hat{L}$ is a 'no' instance. Here is how we do it: for $X \in \hat{L}$ we construct the neural network that solves it by using the reduction to L. We then check every state of the network to see if it is a local maximum (that is done in polynomial time). In case it is a local maximum, we check if the instance is a 'yes' or a 'no' instance (this is also done in polynomial time).
Thus, we have a nondeterministic polynomial time algorithm to recognize any 'no' instance of $\hat{L}$. Thus, the complement of the problem $\hat{L}$ is in NP. But $\hat{L}$ is an NP-complete problem, hence, from Lemma 1 it follows that NP=co-NP. □

Proof of Proposition 2: The result is a corollary of the results in [7], the reader can refer to it for a more complete presentation.
The proof uses the fact that the Restricted Hamiltonian Circuit (RHC) is an NP-complete problem.
Definiton of RHC: Given a graph $G = (V, E)$ and a Hamiltonian path in G. The question is whether there is a Hamiltonian circuit in G?
It is proven in [7] that RHC is NP-complete.

Suppose there exists a polynomial size neural network for finding an ϵ-approximate solution to TSP. Then it can be shown that an instance $X \in$

RHC can be reduced to an instance $\hat{X} \in TSP$, such that in the network $N_{\hat{X}_\epsilon}$ the following holds: if the Hamiltonian path that is given in X corresponds to a local maximum in $N_{\hat{X}_\epsilon}$ then X is a 'no' instance; else, if it does not correspond to a local maximum in $N_{\hat{X}_\epsilon}$ then X is a 'yes' instance. Note that we can check for locality in polynomial time.
Hence, the existence of $N_{\hat{X}_\epsilon}$ for all $\hat{X} \in TSP$ implies that we have a polynomial time algorithm for RHC. □

4 Concluding Remarks

1. In Proposition 1 we let $\mid W \mid$ and $\mid T \mid$ be arbitrary but bounded by a polynomial in the size of a given instance of a problem. If we assume that $\mid W \mid$ and $\mid T \mid$ are fixed for all instances then a similar result to Proposition 1 can be proved without using complexity theory; this result appears in [1].

2. The network which corresponds to TSP, as suggested in [6], can not solve the TSP with guaranteed quality. However, one should note that all the analysis in this paper is a worst case type of analysis. So, it might be that there exist networks that have good behavior on the average.

3. Proposition 1 is general to all NP-hard problems while Proposition 2 is specific to TSP. Both propositions hold for any type of networks in which an iteration takes polynomial time.

4. Clearly, every network has an algorithm which is equivalent to it, but an algorithm does not necessarily have a corresponding network. Thus, if we do not know of an algorithmic solution to a problem we also will not be able to find a network which solves the problem. If one believes that the neural network model is a good model (e.g. it is amenable to implementation with optics), one should develop techniques to program the network to perform an algorithm that is known to have some guaranteed good behavior.

Acknowledgement: Support of the U.S. Air Force Office of Scientific Research is gratefully acknowledged.

References

[1] Y. Abu Mostafa, *Neural Networks for Computing?* in Neural Networks for Computing, edited by J. Denker (AIP Conference Proceedings no. 151, 1986).

[2] J. Bruck and J. Sanz, *A Study on Neural Networks*, IBM Tech Rep, RJ 5403, 1986. To appear in International Journal of Intelligent Systems, 1988.

[3] J. Bruck and J. W. Goodman, *A Generalized Convergence Theorem for Neural Networks and its Applications in Combinatorial Optimization*, IEEE First ICNN, San Diego, June 1987.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[5] J. J. Hopfield, *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, Proc. Nat. Acad. Sci. . USA, Vol. 79, pp. 2554-2558, 1982.

[6] J. J. Hopfield and D. W. Tank, *Neural Computations of Decisions in Optimization Problems*, Biol. Cybern. 52, pp. 141-152, 1985.

[7] C. H. Papadimitriou and K. Steiglitz, *On the Complexity of Local Search for the Traveling Salesman Problem*, SIAM J. on Comp., Vol. 6, No. 1, pp. 76-83, 1977.

[8] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., 1982.

[9] J. C. Picard and H. D. Ratliff, *Minimum Cuts and Related Problems*, Networks, Vol 5, pp. 357-370, 1974.

Neural Networks, Error-Correcting Codes and Polynomials Over the Binary n-Cube

Jehoshua Bruck *
Information Systems Laboratory
EE Department, Stanford University
Stanford, CA 94305

Mario Blaum
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

Abstract

We present several ways of connecting the concept of error-correcting codes with the concept of neural networks. We show that performing maximum likelihood decoding in a linear block error-correcting code is equivalent to finding a global maximum of the energy function of a certain neural network. We also show that given a linear block code we can construct a neural network such that every local maximum of the energy function corresponds to a codeword and every codeword corresponds to a local maximum. We derive a representation theory for boolean functions and use it to extend the results for nonlinear block codes. The connection between maximization of polynomials over the n-cube and error-correcting codes is also investigated; our results suggest that decoding techniques can be a useful tool for solving problems of maximization of polynomials over the n-cube.

*Work done while the author was a summer student and a research student associate at the IBM Almaden Research Center. Partial support of the U.S. Air Force Office of Scientific Research is gratefully acknowledged.

1 Introduction

The main goal in the paper is to explore the connections between the three concepts in the title.

A neural network is a computational model that has recently been attracting a lot of interest because it seems to have properties that are similar to those of both biological and physical systems. The computation that is performed in a neural network is a maximization of a so called energy function. The state space of neural network can be described by the topography which is defined by the energy function associated with the network.

The main problem in the field of error-correcting codes is to design good codes: codes that can correct many errors and whose encoding and decoding procedures are computationally efficient. An error-correcting code can be described by a topography, with the peaks of the topography being the codewords. The decoding of a corrupted word, (a point in the topography which is not a peak) is then equivalent to looking for the closest peak in the topography.

The above analogy between the two subjects was the initial motivation for this work.

It turns out that both neural networks and error correcting codes can be described by polynomials over the n-cube. Thus, the connection between the two concepts can be established. The representation of error correcting codes using polynomials over the n-cube gives also a new perspective of the subject that enables to derive some new proofs for known results.

The problem of maximization of polynomials over the n-cube is a known problem in operations research and computer science. The connection with error correcting codes suggests a new tool for solving these problems, namely, decoding techniques.

The paper is organized as follows: In Section 2, we present some background on neural networks. We review the basic definitions of the Hopfield model. We discuss stable states and the different modes of operation of the network. We conclude the section by proving that finding a global maximum of the energy function of the network is equivalent to finding a minimum cut in a certain graph. The generalization to energy functions of higher degree is also reviewed.

In Section 3, we establish a connection between the Hopfield model and graph theoretic codes. We prove that maximum likelihood decoding in a graph theoretic code is equivalent to finding the minimum cut in a certain graph. By the previous section, this implies that maximum likelihood decoding in a graph theoretic code is equivalent to finding a maximum of the energy in a neural network.

In Section 4, we extend the results of Section 3 to general linear block codes. The key idea is to represent the binary symbols $\{0,1\}$ by the symbols $\{1,-1\}$ with the operation being multiplication instead of exclusive OR. A general energy function, not necessarily quadratic, is defined based on the generator matrix of a given linear block code. We show that finding the global maximum of this energy function is equivalent to maximum likelihood decoding in the code. Some of the results are generalized for finite fields $GF(p)$, p a prime. The idea is to represent the elements as p-roots of unity. For the cases $p = 3$ and $p = 5$, the energy function is generalized. For $p = 3$, maximizing the energy function is equivalent to

maximum likelihood decoding. The same is true for $p = 5$, but with respect to the Lee distance. Several examples with Hamming and first order Reed-Muller codes are given.
In Section 5 we study the energy function associated with the parity check matrix of a code. When this matrix is written in systematic form, we show that each codeword corresponds to a local maximum of the polynomial associated with the parity check matrix, and that each local maximum corresponds to a codeword. We interpret the results of this section as dual of the ones in section 4 for defining the maximum likelihood problem.
In Section 6, several ways of representing boolean functions are discussed. A boolean function is defined as a mapping $f : \{0,1\}^n \rightarrow \{0,1\}$. Given the results of the previous sections, we are interested in representing it with the symbols 1 and -1. We show how to transform a boolean function to an equivalent polynomial over $\{1,-1\}$ and its inverse transform; a boolean function to an equivalent polynomial over $\{0,1\}$ and its inverse transform; and a polynomial over $\{1,-1\}$ to a polynomial over $\{0,1\}$ and its inverse transform. The results are used to generalize the results in Section 4 to nonlinear codes.
In Section 7, we consider the problem of solving unconstrained nonlinear 0-1 programs. This is basically the problem of maximizing a polynomial on n variables, each variable being 0 or 1. It is known that this problem is NP-hard. The known solvable cases use the concept of the conflict graph. We found that the family of polynomials associated with Hamming codes results in a conflict graph which is not bipartite in general (i.e. for which an efficient algorithm is not known). For the family of polynomials associated with Hamming codes efficient recognition and maximization techniques (which are based on decoding techniques) are presented.
A note regarding the notation. Since G denotes a graph (in graph theory) and a generator matrix (in coding theory), we decided to put 'hats' on all notations which are related to graphs. That is, a graph is denoted by $\hat{G} = (\hat{V}, \hat{E})$, while a generator matrix of a code is denoted by G.

2 Background on Neural Networks

The neural network model is a discrete time system that can be represented by a weighted and undirected graph. There is a weight attached to each edge of the graph and a threshold value attached to each node (neuron) of the graph. The *order* of the network is the number of nodes in the corresponding graph. Let N be a neural network of order n: then N is uniquely defined by (W,T) where:

- W is an $n \times n$ symmetric matrix, W_{ij} is equal to the weight attached to edge (i,j).
- T is a vector of dimension n, T_i denotes the threshold attached to node i.

Every node (neuron) can be in one of two possible states, either 1 or -1. The state of node i at time t is denoted by $V_i(t)$. The *state* of the neural network at time t is the vector $V(t)$.

The next state of a node is computed by:

$$V_i(t+1) = sgn(H_i(t)) = \begin{cases} 1 & \text{if } H_i(t) \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

where

$$H_i(t) = \sum_{j=1}^{n} W_{ji} V_j(t) - T_i$$

The next state of the network, i.e. $V(t+1)$, is computed from the current state by performing the evaluation (1) at a subset of the nodes of the network, to be denoted by S. The modes of operation are determined by the method by which the set S is selected in each time interval. If the computation is performed at a single node in any time interval, i.e. $\mid S \mid = 1$, then we will say that the network is operating in a *serial* mode, and if $\mid S \mid = n$ then we will say that that the network is operating in a *fully parallel* mode. All the other cases, i.e. $1 < \mid S \mid < n$ will be called *parallel* modes of operation. The set S can be chosen at random or according to some deterministic rule.
A state $V(t)$ is called *stable* iff $V(t) = sgn(WV(t) - T)$, i.e. there is no change in the state of the network no matter what the mode of operation is. One of the most important properties of the model is its convergence property as summarized by the following proposition.

Proposition 2.1 *[5,7,13] Let $N = (W,T)$ be a neural network, with W being a symmetric matrix Then the network N always converges to a stable state while operating in a serial mode, and to a cycle of length at most 2 while operating in a fully parallel mode.*

The main idea in the proof of the convergence property is to define a so called **energy function** and to show that this energy function is nondecreasing when the state of the network changes as a result of computation. The energy function used in the proof of Proposition 2.1 is:

$$E(t) = V^T(t)WV(t) - 2V^T(t)T \quad (2)$$

A neural network when operating in a serial mode will always get to a stable state which corresponds to a local maximum in the energy function. This suggests the use of the network as a device for performing a local search algorithm for finding a maximal value of the energy function [4,5,14]. Clearly, every optimization problem which can be defined in a form of a quadratic function over $\{-1,1\}^n$ as in (2), can be mapped to a neural network which will perform a search for its optimum. One of the optimization problems which is not only representable by a quadratic function but actually is equivalent to it is the problem of finding the Minimum Cut in a graph [5,18]. In order to make the above statement clear let us start by defining the term cut in a graph.

Definition: Let $\hat{G} = (\hat{V}, \hat{E})$ be a weighted and undirected graph, with W being a symmetric matrix of the weights of the edges of $\hat{G}$. Let $\hat{V}_1$ be a subset of $\hat{V}$, and let $\hat{V}_{-1} = \hat{V} - \hat{V}_1$. The set of edges each of which is incident at a node in $\hat{V}_1$ and at a node in $\hat{V}_{-1}$ is called a *cut* in

$\hat{G}$.

Definition: The weight of a cut is the sum of its edge weights. A *Minimum Cut* (MC) of a graph is a cut with minimum weight.

The equivalence between the MC problem and the problem of maximizing the energy function of a neural network is summarized by the following theorem (generalizations of this equivalence can be found in [4,5]). We include the proof in order to exhibit a principle that will be useful in the sequel.

Proposition 2.2 *[5,18] Let $N = (W,T)$ be a neural network with all thresholds being zero; i.e, $T \equiv 0$. The problem of finding a state V for which the energy E is maximum is equivalent to finding a minimum cut in the graph corresponding to N.*

Proof: Since $T \equiv 0$, the energy function is:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{i,j} V_i V_j \tag{3}$$

Let W^{++} denote the sum of weights of edges in N with both end points equal 1, and let W^{+-} denote the corresponding sums of the other two cases. It follows that:

$$E = 2(W^{++} + W^{--} - W^{+-}) \tag{4}$$

which also can be written as:

$$E = 2(W^{++} + W^{--} + W^{+-}) - 4W^{+-} \tag{5}$$

Since the first term in the above equation is constant (it is the sum of the weights of the edges), it follows that maximization of E is equivalent to the minimization of W^{+-}. It is clear that W^{+-} is the weight of the cut in N with $\hat{V}_1$ being the nodes of N with the state being equal to 1. □

Hence, a neural network operating in a serial mode is equivalent to performing a local search algorithm for finding a minimum cut in the network. Changing the state of a node in the network is equivalent to moving it from one side of the cut to the other in the local search algorithm.

The above definition of the model results in an energy function which is quadratic. The definition of the model can be generalized to energy functions of a higher degree [1]. In the general case, every neuron computes an algebraic threshold function which is equivalent to checking which state (either 1 or -1) will result in a higher value of the energy function.

Example: Consider the energy function:

$$E = W_{1,2,3} V_1 V_2 V_3 + W_{1,2} V_1 V_2 + W_{2,3} V_2 V_3 + W_1 V_1$$

For example, the generalization of (1) for node 1 is:

$$V_1(t+1) = sgn(H_1(t))$$

where

$$H_1(t) = W_{1,2,3}V_2V_3 + W_{1,2}V_2 + W_1$$

We will start by investigating the connections between quadratic energy function and error correcting codes and then continue by looking at general energy functions.

3 Neural Networks and Graph Theoretic Codes

The main goal of this section is to establish the relations between neural networks and graph theoretic error correcting codes. Let us start by defining the family of graph theoretic codes (for more details see [8,17]).
Let $\hat{G} = (\hat{V}, \hat{E})$ be an undirected graph, with $\hat{V}$ being the set of nodes of $\hat{G}$ and $\hat{E}$ being the set of edges of $\hat{G}$. A subset of the set of edges of $\hat{G}$ can be represented by a characteristic vector of length $\mid \hat{E} \mid$, with edge e_i corresponding to the i's entry of the characteristic vector. That is, every $S \subseteq \hat{E}$ can be represented by a vector to be denoted by 1_s; such that:

$$1_s(i) = \begin{cases} 1 & \text{if } e_i \in S \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Definition: The *incident matrix* of a graph $\hat{G} = (\hat{V}, \hat{E})$, to be denoted by $D_{\hat{G}}$, is a $\mid \hat{V} \mid \times \mid \hat{E} \mid$ matrix in which row i is the characteristic vector of the set of edges incident upon node $i \in \hat{V}$.

The following facts from graph theory [3] are the basis for the definition of the family of graph theoretic codes.

Fact 1: The set of characteristic vectors which corresponds to the cuts in a connected graph $\hat{G} = (\hat{V}, \hat{E})$ forms a linear vector space over $GF(2)$, with dimension $(\mid \hat{V} \mid -1)$. The linear vector space that corresponds to the cuts of a graph $\hat{G}$ will be denoted as the *cut space* of $\hat{G}$.
It is interesting to note that the circuits in a graph constitute also a linear vector space.

Fact 2: Given a connected graph $\hat{G} = (\hat{V}, \hat{E})$, the incident matrix of $\hat{G}$ has rank $\mid \hat{V} \mid -1$. Every row in $D_{\hat{G}}$ is a characteristic vector of a cut, and every $\mid \hat{V} \mid -1$ rows of $D_{\hat{G}}$ form a basis for the cut space of $\hat{G}$.

Hence, given a connected graph $\hat{G}$, the cut space of the graph is a *linear block code* [15,17] of dimension $\mid \hat{V} \mid -1$; thus, every graph has an $[\mid \hat{E} \mid, \mid \hat{V} \mid -1]$ code associated with its cuts. The code associated with the cuts of a graph $\hat{G}$ will be denoted by $C_{\hat{G}}$.
The codes associated with graphs, that is, cut codes and circuit codes, are called **graph**

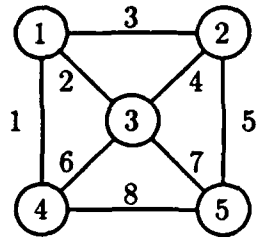

Figure 1: A graph which corresponds to an (8,4) code.

theoretic codes. In this paper only cut codes will be discussed.

Example: Let $\hat{G}$ be the graph in Figure 1, $\hat{G}$ has 5 nodes and 8 edges. The incident matrix of the graph $\hat{G}$ is:

$$D_{\hat{G}} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \tag{7}$$

Any 4 rows of $D_{\hat{G}}$ form a basis of the cut space of $\hat{G}$. For example, the matrix which consists of the first 4 rows of $D_{\hat{G}}$ is a generator matrix of the error correcting linear block code associated with $\hat{G}$. It is easily observed that $\hat{G}$ does not contain a cut with less than 3 edges (besides the empty cut); thus, the code $C_{\hat{G}}$ has minimum distance 3 and can correct one error.

Given a graph $\hat{G}$, an interesting question is, how to formulate the Maximum Likelihood Decoding (MLD) problem of the code $C_{\hat{G}}$ in a graph theoretic language. That is, given a graph $\hat{G} = (\hat{V}, \hat{E})$, and a vector Y in $\{0,1\}^{|\hat{E}|}$, what is the codeword in $C_{\hat{G}}$ that is the closest to Y in Hamming distance. The following lemma will answer this question.

Lemma 3.1 *Let $\hat{G} = (\hat{V}, \hat{E})$ be a connected graph. Let $C_{\hat{G}}$ be the code associated with $\hat{G}$. Let Y be a vector over $\{0,1\}^{|\hat{E}|}$. Construct a new graph, to be denoted by $\hat{G}_Y$, by assigning weights to the edges of $\hat{G}$ as follows:*

$$W_i = (-1)^{Y_i} \tag{8}$$

W_i is the weight associated with edge i in $\hat{G}$.
Then MLD of Y with respect to $C_{\hat{G}}$ is equivalent to finding the minimum cut in $\hat{G}_Y$.

Proof: Let us assume that Y contains a 1's. Let M be an arbitrary codeword in $C_{\hat{G}}$. Let $N^{i,j}$ denote the number of positions in which M contains an $i \in \{0,1\}$ and Y contains a $j \in \{0,1\}$. Clearly,

$$a = N^{0,1} + N^{1,1}$$

Thus,

$$-N^{1,1} + N^{1,0} = N^{0,1} - a + N^{1,0} \tag{9}$$

Minimizing the right hand side in equation (9) over all $M \in C_{\hat{G}}$ is equivalent to finding a codeword which is the closest to Y. On the other hand, minimizing the left hand side is equivalent to finding the minimum cut in $\hat{G}_Y$. □

From Lemma 3.1 it follows that:

Theorem 3.1 *Let $\hat{G} = (\hat{V}, \hat{E})$ be a connected graph. Then MLD of a word Y with respect to $C_{\hat{G}}$ is equivalent to finding the maximum of the energy function E of the neural network which is defined by the graph $\hat{G}_Y$ and all its threshold values are equal to 0.*

Proof: By Lemma 3.1: MLD of Y with respect to $C_{\hat{G}}$ is equivalent to finding the minimum cut in $\hat{G}_Y$. By Proposition 2.2: Finding the minimum cut in a graph is equivalent to finding the maximum of the energy function of a neural network defined by a graph with all thresholds being zero. □

Graph theoretic error correcting codes are limited in the sense that [8]:

$$d^* \leq \frac{2 \mid \hat{E} \mid}{\mid \hat{V} \mid} \tag{10}$$

where d^* is the minimum distance of the code. For example, a $[7, 4]$ Hamming code is not a graph theoretic code because it has minimum distance 3, and $\frac{14}{5} < 3$. Hence, an interesting question is whether the equivalence stated by Theorem 3.1 can be generalized to all linear block codes. The energy function associated with the MLD of graph theoretic codes is quadratic. It turns out that the energy function associated with the MLD of a general linear block code is a polynomial over the n-cube. The discussion regarding the generalization is the subject of Section 4.

4 Error-Correcting Codes and Energy Functions

In this section we will extend the results in Section 3 and show that the MLD problem of linear block codes is equivalent to maximization of polynomials over the binary n-cube. It will be also shown that the results can be generalized to non-binary codes.

4.1 The Binary Case

Consider a binary linear $[n, k]$ error-correcting block code to be denoted by C [15,17]. The code C is defined by a $k \times n$ generator matrix G. An information vector $b = (b_1, b_2, \ldots, b_k)$ is encoded into the codeword $v = (v_1, v_2, \ldots, v_n)$ such that:

$$v_j = \bigoplus_{i=1}^{k} b_i g_{i,j} \qquad 1 \leq j \leq n$$

where $\bigoplus$ denotes Exclusive OR.

The key idea in the derivation is to represent the symbols of the additive group Z_2 as symbols in the multiplicative group $\{1,-1\}$ through the transformation

$$a \rightarrow (-1)^a$$

i.e.,

$$0 \rightarrow 1, \quad 1 \rightarrow -1$$

We will use a different notation for the $\{1,-1\}$ representation: The information vector $b = (b_1, b_2, \ldots, b_k)$ is represented as $x = (x_1, x_2, \ldots, x_k)$, where $x_i = (-1)^{b_i}$, and the encoded codeword $v = (v_1, v_2, \ldots, v_n)$ is represented as $y = (y_1, y_2, \ldots, y_n)$. Hence,

$$y_j = (-1)^{v_j} = (-1)^{\bigoplus_{i=1}^{k} b_i g_{ij}} = \prod_{i=1}^{k} (-1)^{b_i g_{ij}} = \prod_{i=1}^{k} x_i^{g_{ij}} \tag{11}$$

Example: Consider the $[7,4]$ systematic Hamming code whose generator matrix is given by

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Given the 4 information symbols (b_1, b_2, b_3, b_4), the corresponding codeword is

$$v = (b_1, b_2, b_3, b_4, b_2 \oplus b_3 \oplus b_4, b_1 \oplus b_3 \oplus b_4, b_1 \oplus b_2 \oplus b_4)$$

In the $\{1,-1\}$ representation, this looks like

$$y = (x_1, x_2, x_3, x_4, x_2x_3x_4, x_1x_3x_4, x_1x_2x_4)$$

where $x_j = (-1)^{b_j}$.

Definition: In the $\{1,-1\}$ representation of a code instead of a generator matrix, given an information vector $x = (x_1, x_2, \ldots, x_k)$, we will use an *encoding procedure* $x \rightarrow y$, where $y = (y_1, y_2, \ldots, y_n)$ and $y_j = y_j(x_1, x_2, \ldots, x_k)$ is a monomial. An encoding procedure is *systematic* iff $y_j(x_1, x_2, \ldots, x_k) = x_j$ for $1 \leq j \leq k$.

In the example, the $[7,4]$ Hamming code is described by the systematic encoding procedure:

$$(x_1, x_2, x_3, x_4) \rightarrow (x_1, x_2, x_3, x_4, x_2x_3x_4, x_1x_3x_4, x_1x_2x_4) \tag{12}$$

Another example, the first order (shortened) Reed-Muller code $R(1,3)$ [15] is described by the systematic encoding procedure:

$$(x_1, x_2, x_3) \rightarrow (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3)$$

while the first order Reed-Muller code $R(1,3)$ is described by the encoding procedure:

$$(x_0, x_1, x_2, x_3) \rightarrow (x_0, x_0x_1, x_0x_2, x_0x_1x_2, x_0x_3, x_0x_1x_3, x_0x_2x_3, x_0x_1x_2x_3) \tag{13}$$

The generalization to any $R(1,m)$ first order Reed Muller code is obvious.

Definition: Let G be a $k \times n$ matrix of 1's and 0's. The *polynomial representation* of G with respect to a vector $\omega \in \{1,-1\}^n$, to be denoted by E_ω, is:

$$E_\omega(x) = \sum_{j=1}^{n} \omega_j \prod_{i=1}^{k} x_i^{g_{i,j}} \tag{14}$$

Consider the linear block code defined by the generator matrix G (or equivalently by the encoding procedure associated with G). The polynomial representation of G, i.e. $E_\omega(x)$, will be called the *energy function* of ω with respect to the encoding procedure $x \rightarrow y$. Note that $E_\omega(x) = \omega \cdot y(x)$.
To establish the connection between energy functions and linear block codes, we will prove that finding the global maximum of $E_\omega(x)$ is equivalent to MLD of a vector ω with respect to the code C.

Theorem 4.1 *Given an $[n,k]$ code C defined by an encoding procedure $x \rightarrow y$, and a vector $\omega \in \{1,-1\}^n$, the closest codeword (in Hamming distance) to ω in C corresponds to an information vector $b = (b_1, b_2, \ldots, b_k)$ if and only if*

$$E_\omega(b) = \max_{x \in \{1,-1\}^k} E_\omega(x)$$

Proof: Notice that for any information vector $x \in \{1,-1\}^k$,

$$\begin{aligned} E_\omega(x) &= \textstyle\sum_{j=1}^{n} \omega_j y_j(x) \\ &= \mid \{j : \omega_j = y_j(x)\} \mid - \mid \{j : \omega_j \neq y_j(x)\} \mid \\ &= n - 2 \mid \{j : \omega_j \neq y_j(x)\} \mid \\ &= n - 2d_H(\omega, y) \end{aligned}$$

where d_H denotes Hamming distance. This expression implies that $E_\omega(b_1, b_2, \ldots, b_k)$ will achieve a maximum iff $d_H(\omega, y)$ achieves a minimum. □

Example: Consider the $[7,4]$ Hamming code, defined by the encoding procedure in (12). Assume we want to perform MLD of the received word

$$\omega = (1,-1,-1,1,1,-1,1)$$

Then,

$$E_\omega(x_1, x_2, x_3, x_4) = x_1 - x_2 - x_3 + x_4 + x_2x_3x_4 - x_1x_3x_4 + x_1x_2x_4$$

The maximum of this polynomial occurs at $E_\omega(1, -1, -1, 1) = 5$. So, the received word is decoded as $(1, -1, -1, 1)$.

Example: Consider the $R(1,3)$ first order Reed-Muller code, defined by the encoding procedure in (13). Given the received word $\omega = (\omega_0, \omega_1, \ldots, \omega_7)$, the energy function is

$$\begin{aligned} E_\omega(x_0, x_1, x_2, x_3) &= x_0(\omega_0 + \omega_1x_1 + \omega_2x_2 + \omega_3x_1x_2 + \omega_4x_3 + \omega_5x_1x_3 + \omega_6x_2x_3 + \omega_7x_1x_2x_3) \\ &= x_0(\omega_0 + E_\omega(x_1, x_2, x_3)) \end{aligned}$$

where

$$E_\omega(x_1, x_2, x_3) = \omega_1x_1 + \omega_2x_2 + \omega_3x_1x_2 + \omega_4x_3 + \omega_5x_1x_3 + \omega_6x_2x_3 + \omega_7x_1x_2x_3$$

Hence, it is enough to find

$$\max_{x_1,x_2,x_3 \in \{1,-1\}} | E_\omega(x_1, x_2, x_3) |$$

If the energy that corresponds to the maximum is positive, then $x_0 = 1$, otherwise, $x_0 = -1$. Assume we receive $\omega = (-1, 1, 1, -1, 1, 1, 1, -1)$. We have,

$$E_\omega(x_1, x_2, x_3) = x_1 + x_2 - x_1x_2 + x_3 + x_1x_3 + x_2x_3 - x_1x_2x_3$$

then

$$\max_{x_1,x_2,x_3 \in \{1,-1\}} | E_\omega(x_1, x_2, x_3) | = E_\omega(1, 1, 1) = 3$$

Since the energy is positive, the received word is decoded as $(1, 1, 1, 1)$. In this case the decoding is not unique, since the maximum is achieved at more than one point.

Given an encoding procedure, we can use the same argument as in Theorem 4.1 to determine the minimum distance of the code.
Consider the encoding procedure

$$x = (x_1, x_2, \ldots, x_k) \rightarrow y = (y_1, y_2, \ldots, y_n)$$

and the energy function with $\omega = (1, 1, \ldots, 1)$

$$E(x_1, x_2, \ldots x_k) = y_1 + y_2 + \cdots + y_n$$

As before,

$$E(x_1, x_2, \ldots x_k) = n - 2d_H((11\ldots1), (y_1, y_2, \ldots, y_n))$$

and

$$\min_{(x_1,x_2,\ldots,x_k) \neq (11\ldots1)} d_H((11\ldots1), (y_1, y_2, \ldots, y_n))$$

occurs at

$$M \stackrel{\text{def}}{=} \max_{(x_1,x_2,\ldots x_k)\neq(11\ldots1)} E(x_1, x_2, \ldots, x_k)$$

So, the conclusion is that d^* (the minimum distance of the code) is given by

$$d^* = \frac{n - M}{2} \tag{15}$$

Example: For the $[7, 4]$ Hamming code,

$$M = \max_{(x_1,x_2,x_3,x_4)\neq(1111)} x_1 + x_2 + x_3 + x_4 + x_2x_3x_4 + x_1x_3x_4 + x_1x_2x_4 = 1$$

so

$$d^* = \frac{n - M}{2} = \frac{7 - 1}{2} = 3$$

Example: For the $R(1,3)$ first order Reed-Muller code,

$$M = \max_{(x_0,x_1,x_2,x_3)\neq(1111)} x_0(1 + x_1 + x_2 + x_1x_2 + x_3 + x_1x_3 + x_2x_3 + x_1x_2x_3)$$

which can be written as,

$$M = \max_{(x_0,x_1,x_2,x_3)\neq(1111)} x_0(1 + x_1)(1 + x_2)(1 + x_3) \tag{16}$$

The maximum in (16) is $M = 0$, because at least one of the x_i's (for $i > 1$) must be equal to -1. Thus,

$$d^* = \frac{n - M}{2} = \frac{8}{2} = 4$$

The same argument can be used for any $R(1, m)$ code, giving

$$d^* = \frac{2^m}{2} = 2^{m-1}$$

4.2 Generalization to Non-Binary Codes

Consider now a linear $[n, k]$ error-correcting code over a field $GF(p)$, p a prime. Let G be the generator matrix of the code. Then k symbols in Z_p $(b_1, b_2, \ldots, b_k)$ are encoded into codeword $v = (v_1, v_2, \ldots, v_n)$ as follows,

$$v_j = \sum_{m=1}^{k} b_m g_{mj} (\mathrm{mod} p), \qquad 1 \leq j \leq n$$

Again, the key idea is to use the multiplicative representation: let u be the pth root of unity,

$$u = e^{\frac{2\pi i}{p}}$$

The additive group Z_p can be represented as a multiplicative group of p-roots of unity through the transformation $a \rightarrow u^a$.
In the multiplicative representation, the k information symbols $(b_1, b_2, \ldots, b_k)$ are represented as

$$(x_1, x_2, \ldots, x_k) = (u^{b_1}, u^{b_2}, \ldots, u^{b_k})$$

so the encoded codeword $v = (v_1, v_2, \ldots, v_n)$ is represented as $y = (y_1, y_2, \ldots, y_n)$ where

$$y_j = u^{v_j} = u^{\sum_{m=1}^{k} b_m g_{mj} (\mathrm{mod} p)} = \prod_{m=1}^{k} u^{b_m g_{mj}} = \prod_{m=1}^{k} x_m^{g_{mj}}$$

Example: Consider the [4, 2] ternary Hamming code whose generator matrix is

$$G = \begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 2 & 1 \end{pmatrix}$$

Given the two information symbols (b_1, b_2), the corresponding codeword is

$$v = (b_1, b_2, 2b_1 + 2b_2 (\mathrm{mod} 3), b_2)$$

In the multiplicative representation, this becomes

$$y = (x_1, x_2, x_1^2 x_2^2, x_2)$$

where $x_j = u^{b_j}$, $u = e^{\frac{2\pi i}{3}}$.

Hence, as for the binary case, we can represent a code on a field with p elements (p a prime) by an encoding procedure. The elements are now p roots of unity. So, given k information symbols, we have the 1-1 assignment

$$x = (x_1, x_2, \ldots, x_k) \rightarrow y = (y_1, y_2, \ldots, y_n)$$

where $y_j = y_j(x_1, x_2, \ldots, x_k)$ is a monomial.

We will show that for $p = 3$ or 5, there are easy expressions of the energy function that generalize the binary case ($p = 2$).
We start by redefining the energy function. Given an encoding procedure

$$x = (x_1, x_2, \ldots, x_k) \rightarrow y = (y_1, y_2, \ldots, y_n)$$

and $\omega = (\omega_1, \omega_2, \ldots, \omega_n)$ a vector whose entries are pth roots of unity, we define the energy function as follows:

$$E_\omega(x) = \lfloor \Re(\bar{\omega}_1 y_1) \rfloor + \lfloor \Re(\bar{\omega}_2 y_2) \rfloor + \cdots + \lfloor \Re(\bar{\omega}_n y_n) \rfloor$$

where $\Re(x)$ denotes real part, $\lfloor x \rfloor$ integer part and $\bar{x}$ complex conjugate of x.
Notice that this energy function coincides with the one in [2] (in that case, $u = -1$).
Before proceeding further, let's recall the definition of the *Lee distance* [17].

Definition: The *Lee weight* of an $n-$tuple $(a_1, a_2, \ldots, a_n)$, $a_i \in \mathcal{Z}_p$, p a prime, is defined as

$$w_L = \sum_{i=1}^{n} |a_i|$$

where

$$|a_i| = \begin{cases} a_i & 0 \leq a_i \leq \frac{p}{2} \\ p - a_i & \frac{p}{2} < a_i \leq p-1 \end{cases}$$

The *Lee distance* between two $n-$tuples is defined as the Lee weight of their difference.

We study the cases $p = 3$ and $p = 5$. From now on, $x \rightarrow y$ denotes an encoding procedure that defines a code C, and x and y are vectors of length k and n, respectively, of 3 or 5 roots of unity.
We are going to prove two theorems. The first one is similar to Theorem 4.1. It states that MLD in a ternary code is equivalent to the maximization of the energy function in (17). The second theorem states something similar for codes on the 5 roots of unity, but with respect to the Lee distance.

Theorem 4.2 *Let $p = 3$, $a \rightarrow b$, then b is the closest codeword (in Hamming distance) to a word ω if and only if*

$$E_\omega(a) = \max_x E_\omega(x)$$

Proof: Similar to that of Theorem 4.1. □

Example: Consider again the $[4,2]$ ternary Hamming code. Assume $\omega = (u, u^2, 1, u)$ is received ($u = e^{\frac{2\pi i}{3}}$), then

$$E_\omega(x_1, x_2) = \lfloor \Re(u^2 x_1) \rfloor + \lfloor \Re(u x_2) \rfloor + \lfloor \Re(x_1^2 x_2^2) \rfloor + \lfloor \Re(u^2 x_2) \rfloor$$

It can be easily verified that $\max E_\omega(x_1, x_2) = E_\omega(u, u^2) = 2$, so ω is decoded as (u, u^2).

Theorem 4.3 *Let $p = 5$, $a \rightarrow b$, then b is the closest codeword (in Lee distance) to a word ω if and only if*

$$E_\omega(a) = \max_x E_\omega(x)$$

Proof: Using the definition of the energy function,

$$\begin{aligned} E_\omega(a) &= \mid \{j : \bar{\omega}_j b_j = 1\} \mid - \mid \{j : \bar{\omega}_j b_j = \omega^2 \text{ or } \omega^3\} \mid \\ &= n - \mid \{j : \bar{\omega}_j b_j = \omega \text{ or } \omega^4\} \mid - 2 \mid \{j : \bar{\omega}_j b_j = \omega^2 \text{ or } \omega^3\} \mid \\ &= n - d_L(\omega, b) \end{aligned}$$

where d_L denotes Lee distance.
Hence, $E_\omega(a)$ reaches a maximum if and only if $d_L(\omega, b)$ reaches a minimum. □

Example: Consider the [6,2] code on Z_5 generated by

$$G = \begin{pmatrix} 1 & 2 & 3 & 4 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

The corresponding encoding procedure, taking the symbols as 5 roots of unity, is given by

$$(x_1, x_2) \rightarrow (x_1 x_2, x_1^2 x_2, x_1^3 x_2, x_1^4 x_2, x_1, x_2)$$

Assume $\omega = (u^2, u^4, 1, u^3, u, 1)$ is received, where $u = e^{\frac{2\pi i}{5}}$. The energy function is then

$$E_\omega(x_1, x_2) = \lfloor \Re(u^3 x_1 x_2) \rfloor + \lfloor \Re(u x_1^2 x_2) \rfloor + \lfloor \Re(x_1^3 x_2) \rfloor + \lfloor \Re(u^2 x_1^4 x_2 \rfloor + \lfloor \Re(u^4 x_1) \rfloor + \lfloor \Re(x_2) \rfloor$$

It can be verified that the maximum occurs at $E_\omega(u^2, 1) = 4$, so ω is decoded as $(u^2, 1)$.

5 Representing Linear Codes as Stable States of Energy Functions

Let $\mathcal{C}$ be a linear block code (over $GF(2)$) defined by the generator matrix G. Let E_C be a polynomial over $\{1, -1\}$ (energy function) with the property that every local maximum in E_C corresponds to a codeword in $\mathcal{C}$ and every codeword in $\mathcal{C}$ corresponds to a local maximum in E_C.
Consider the following question: given a code $\mathcal{C}$ defined by G, is there an efficient algorithm to construct E_C? This section describes the development of such an algorithm.

Consider the $[n, k]$ linear block code $\mathcal{C}$. Without loss of generality, let us assume that the generator matrix G is given in a systematic form; that is,

$$G = [I_k : P] \tag{18}$$

where I_k is a $k \times k$ identity matrix, and P is a $k \times (n-k)$ matrix. The parity check matrix of $\mathcal{C}$ is:

$$H^T = \begin{bmatrix} P \\ \cdots \\ I_{n-k} \end{bmatrix} \tag{19}$$

By the definition of H, for all $X \in \mathcal{C}$,

$$XH^T = \mathbf{0} \tag{20}$$

where $\mathbf{0}$ in (20) is an all zero vector of length $(n-k)$. Equation (20) can be written using the polynomial representation devised in (14), with the vector of coefficients being the all-1 vector.

Lemma 5.1 *Let $\tilde{E}(X)$ be the polynomial representation of H^T with respect to the all-1 vector. Then $X \in \mathcal{C}$ iff $\tilde{E}(X) = n-k$.*

Proof: $\tilde{E}$ has $(n-k)$ terms, and all the coefficients are equal 1. Hence, $\tilde{E} = n-k$ iff all the terms are equal to 1. □

The lemma ensures that in the polynomial $\tilde{E}$ every codeword corresponds to a global maximum (stable state). But does every local maximum correspond to a codeword?

Theorem 5.1 *Let $\mathcal{C}$ be a linear block code, with G,H,E_C and $\tilde{E}$ as defined above; then $\tilde{E}$ is a polynomial with the properties of E_C. That is, X corresponds to a local maximum in $\tilde{E}$ iff $X \in \mathcal{C}$.*

Proof: One direction follows from the lemma. The global maximum of $\tilde{E}$ is $n-k$; thus, every codeword is a global (and a local) maximum.
The second direction follows from the fact that H has a systematic form. The last $n-k$ variables in $\tilde{E}$; that is, $X_{k+1}, \ldots, X_n$, appear only in one term each. That is, X_{k+1} appears only in the first term, X_{k+2} appears only in the second term and so on. Assume there exists a vector V which corresponds to a local maximum (which is not global). That is $\tilde{E}(V) = L$, where $L < n-k$. Hence, there exists at least one term in $\tilde{E}(V)$ which is not 1. But this term can be made 1 by flipping the value of the variable which appears only in this term. This contradicts the fact that V is a local maximum. □

Examples:

1. Consider the single parity check code, it is an $[n, n-1]$ code and

$$G = [I_{n-1} : 1_{n-1}]$$

$$H^T = 1_n$$

where 1_n is the all-1 vector of length n. Hence,

$$\tilde{E}(X) = X_1 X_2 \cdots X_n$$

It is clear that $\tilde{E}(X) = 1$ iff $X \in \mathcal{C}$. Also, $\tilde{E}(X) = -1$ for all $X \notin \mathcal{C}$. Thus, local maxima in $\tilde{E}$ have one to one correspondence with codewords in $\mathcal{C}$

2. Consider the simple repetition code, it is an $[n, 1]$ code and

$$G = [1, 1, \ldots, 1]$$

$$H = [1_{n-1} : I_{n-1}]$$

And

$$\tilde{E}(X) = X_1(X_2 + X_3 + \ldots + X_n)$$

It is clear that there are two stable states in $\tilde{E}$ above, the all-1 and the all-(-1) vectors.

3. Consider the $[7, 4]$ Hamming code (see also Section 4),

$$H^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{21}$$

$$\tilde{E}(X) = X_1X_2X_4X_5 + X_1X_3X_4X_6 + X_2X_3X_4X_7 \tag{22}$$

Again, the polynomial in (22) has the $[7, 4]$ Hamming code as the set of its local maxima.

To summarize, given a linear code $\mathcal{C}$ the algorithm for constructing a polynomial E_C is as follows:

1. Construct the systematic generator matrix of $\mathcal{C}$ by performing row operation on the generator matrix G.
2. Construct the systematic parity check matrix of $\mathcal{C}$, according to (19).
3. Construct $\tilde{E}$, which is the polynomial representation of H^T with respect to the all-1 vector. By Theorem 5.1, let $E_C = \tilde{E}$.

A few remarks and generalizations regarding the above development:

1. The construction described above also works for cosets of linear codes. Let ω be the vector of length $n - k$ of the coefficients of $\tilde{E}$. In the above construction we chose ω to be the all-1 vector, and got that $E_C = \tilde{E}$. Let $\hat{\mathcal{C}}$ be a coset of $\mathcal{C}$, and let S be the syndrome which corresponds to $\hat{\mathcal{C}}$. It can be proven (basically as in the proof of Theorem 5.1) that there is a 1-1 correspondence between local maxima of the polynomial representation of H^T with $\omega = S$ and the vectors in the coset $\hat{\mathcal{C}}$.
Clearly, the syndrome which corresponds to the code $\mathcal{C}$ is the all-1 vector (remembering that in the transformation in Section 4, 0 goes to 1).

4. The above construction (with respect to the one suggested in Section 4) is a dual way of defining the MLD problem. Consider a linear block code $\mathcal{C}$, defined by its parity check matrix H. Given a vector V, the MLD problem can be defined as finding the local maximum in E_C which is the closest to V. Or equivalently, finding a local maximum of the energy function associated with the syndrome (corresponding to V) that is achieved by a vector of minimum weight.

6 Boolean Functions, Polynomials and Codes

In this section the representation of boolean functions as polynomials over the field of real numbers is investigated. In view of the results in Section 4, applications of the derived representation to coding theory are also investigated. Although a part of the material in this section is known (see for example [15,16]), we include the detailed derivation as we believe that it is novel with regard to the mode of presentation. Let us start by some definitions and notations.

Definition: A boolean function f on n variables, is a mapping,

$$f : \{0,1\}^n \longrightarrow \{0,1\}$$

As in section 4, it is useful to define boolean functions using the symbols '1' and '-1' instead of using the symbols '0' and '1', respectively.

Definition: A *Hadamard matrix* of order m, to be denoted by H_m, is an $m \times m$ matrix of +1's and -1's such that:

$$H_m H_m^T = m I_m \tag{23}$$

where I_m is the $m \times m$ identity matrix. The above definition is equivalent to saying that any two rows of H are orthogonal.

Hadamard matrices of order 2^k exist for all $k \geq 0$. The so called Sylvester construction is as follows:

$$\begin{aligned} H_1 &= [1] \\ H_2 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ H_{2^{n+1}} &= \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix} \end{aligned} \tag{24}$$

Definition: Given a boolean function f of order n, P_f is a polynomial (with coefficients over the field of real numbers) *equivalent* to f iff for all $X \in \{1,-1\}^n$:

$$f(X) = P_f(X)$$

Problem: Given a boolean function f of order n, compute P_f - a polynomial which is equivalent to f.

As an example, let $f = x_1 \oplus x_2$; that is, f is the XOR function of two variables. It is easy to check that in the $\{1, -1\}$ representation $P_f = x_1 x_2$.
Notice that for every boolean function f, the polynomial P_f is linear in each of its variables because $x^2 = 1$ for $x \in \{-1, 1\}$. It turns out that every boolean function has a unique representation as a polynomial. This representation is derived by using the Hadamard matrix, as described by the following theorem.

Theorem 6.1 *Let f be a boolean function of order n. Let P_f be a polynomial equivalent to f. Let A denote the vector of coefficients of P_f. Let P denote the vector of the 2^n values of P_f (and f). Then:*

1. *The polynomial P_f always exists and is unique.*
2. *The coefficients of P_f are computed as follows,*

$$A = \frac{1}{2^n} H_{2^n} P$$

Proof: The proof is constructive. The idea is to compute A by solving a system of linear equations. Let us start by computing the coefficients of P_f, for f being a function of one variable.

$$P_f = a_0 + a_1 x_1$$

and,

$$\begin{aligned} P_f(1) &= a_0 + a_1 \\ P_f(-1) &= a_0 - a_1 \end{aligned}$$

clearly,

$$P = H_2 A \tag{25}$$

and by (23),

$$A = \frac{1}{2} H_2 P \tag{26}$$

Claim: The above result can be generalized to n variables as follows:

$$P = H_{2^n} A \tag{27}$$

The proof is by induction. The case $n = 1$ was proven above. Assume (27) is true for n. Clearly, every polynomial of $n + 1$ variables can be written as a combination of two polynomials of n variables each,

$$P_f(x_1, .., x_{n+1}) = P_f^1(x_1, .., x_n) + x_{n+1} P_f^2(x_1, ..x_n) \tag{28}$$

x_3	x_2	x_1	f
1	1	1	1
1	1	−1	1
1	−1	1	1
1	−1	−1	−1
−1	1	1	1
−1	1	−1	−1
−1	−1	1	1
−1	−1	−1	−1

Figure 2: the truth table of f.

There are two possibilities, either $x_{n+1} = 1$ or $x_{n+1} = -1$. Hence, by the induction hypothesis (27), the system of linear equations for $n+1$ variables becomes:

$$P = \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix} A \tag{29}$$

Following from the recursive definition of Hadamard matrices (24),

$$P = H_{2^{n+1}} A \tag{30}$$

Hadamard matrices are nonsingular; thus, for any given f a unique P_f (defined by the vector of coefficients A) always exists. □

Example: Consider the function f of 3 variables,

$$f = (x_1 \wedge x_3) \vee (x_1 \wedge x_2) \tag{31}$$

The truth table of f appears in Figure 2 (note that a logical 0 is mapped to 1, and a logical 1 is mapped to -1). By Theorem 6.1,

$$P_f = \frac{1}{8}(2 + 6x_1 + 2x_2 - 2x_1x_2 + 2x_3 - 2x_1x_3 + 2x_2x_3 - 2x_1x_2x_3) \tag{32}$$

A few remarks concerning the above method:

1. Special care should be taken with respect to the the order by which the values of f are specified in P. The function f should be specified according to the natural order with the highest index being the most significant bit (as in Figure 2).

2. A monomial can be described by a vector of 1's and -1's with a variable appearing in the monomial iff it corresponds to -1. For example, $(-1, -1, 1)$ corresponds to x_3x_2. Using this this description, the monomials of P_f appear according to the natural order with the highest index being the most significant bit.
   In (32) the terms are written according the order they appear in A for 3 variables. This order will be denoted as the *natural order*.

3. The above method is applicable not just to boolean functions but to any function of the form $f : \{1,-1\}^n \longrightarrow \Re$.

The representation theory developed above holds also if one is interested in the question of finding an equivalent polynomial, over $\{0,1\}$, of a boolean function. To see this, simply observe that any monomial over $\{(1,-1\}$ can be written as a polynomial over $\{0,1\}$ by the change of variable $x = 1 - 2u$, as follows:

$$\prod_{i=1}^{k} x_i = 1 + \sum_{i=1}^{k} (-2)^i \sum_{S_i} \prod_{j \in S_i} u_j \tag{33}$$

with S_i being a subset of $\{1,\ldots,k\}$ with i elements.
For example,

$$x_1x_2x_3 = 1 - 2(u_1 + u_2 + u_3) + 4(u_1u_2 + u_1u_3 + u_2u_3) - 8u_1u_2u_3$$

The question is: what is the form of the transformation matrix from a boolean function to its equivalent 0-1 polynomial? To answer this question it is useful to use the same technique as in Theorem 6.1. That is, to define the transformation recursively.

Lemma 6.1 *Let A be the vector of the coefficients of a polynomial over $\{1,-1\}^n$, with the coefficients ordered according to the so called natural order. Let $\hat{A}$ be the the vector of the coefficients of a polynomial over $\{0,1\}^n$ which is equal to the polynomial associated with A. Then*

$$\hat{A} = F_{2^n} A$$

where F_{2^n} is defined recursively as follows:

$$\begin{aligned} F_1 &= [1] \\ F_2 &= \begin{bmatrix} 1 & 1 \\ 0 & -2 \end{bmatrix} \\ F_{2^{n+1}} &= \begin{bmatrix} F_{2^n} & F_{2^n} \\ 0 & -2F_{2^n} \end{bmatrix} \end{aligned} \tag{34}$$

And also,

$$\begin{aligned} F_1^{-1} &= [1] \\ F_2^{-1} &= \begin{bmatrix} 1 & 0.5 \\ 0 & -0.5 \end{bmatrix} \\ F_{2^{n+1}}^{-1} &= \begin{bmatrix} F_{2^n}^{-1} & 0.5F_{2^n}^{-1} \\ 0 & -0.5F_{2^n}^{-1} \end{bmatrix} \end{aligned} \tag{35}$$

Proof: The proof is by induction, using the same arguments as in Theorem 6.1. □

Using the above lemma, we can formulate the following theorem which is the equivalent of Theorem 6.1 for the $\{0,1\}$ case.

Theorem 6.2 *Let f be a boolean function of order n. Let $\hat{P}_f$ be a polynomial over $\{0,1\}^n$ which is equivalent to f. Then a unique $\hat{P}_f$ always exists and is computed as described in the following proof.*

Proof: The existence and uniqueness of $\hat{P}_f$ follows from Theorem 6.1. By Theorem 6.1 and Lemma 6.1,

$$\hat{A} = F_{2^n} H_{2^n}^{-1} \hat{P} \tag{36}$$

Let

$$\hat{H}_{2^n} \stackrel{\text{def}}{=} F_{2^n} H_{2^n}^{-1}$$

Then, by the recursive definition of F and H,

$$\hat{H}_{2^{n+1}} = \frac{1}{2} \begin{bmatrix} F_{2^n} & F_{2^n} \\ 0 & -2F_{2^n} \end{bmatrix} \begin{bmatrix} H_{2^n}^{-1} & H_{2^n}^{-1} \\ H_{2^n}^{-1} & -H_{2^n}^{-1} \end{bmatrix} \tag{37}$$

Performing the multiplication above in blocks results in,

$$\hat{H}_{2^{n+1}} = \begin{bmatrix} F_{2^n} H_{2^n}^{-1} & 0 \\ -F_{2^n} H_{2^n}^{-1} & F_n H_{2^n}^{-1} \end{bmatrix} \tag{38}$$

Thus, the recursive definition of $\hat{H}$ is as follows:

$$\begin{aligned} \hat{H}_1 &= [1] \\ \hat{H}_2 &= \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \\ \hat{H}_{2^{n+1}} &= \begin{bmatrix} \hat{H}_{2^n} & 0 \\ -\hat{H}_{2^n} & \hat{H}_{2^n} \end{bmatrix} \end{aligned} \tag{39}$$

□

The inverse of $\hat{H}$ can be derived by using the recursive definition of $\hat{H}$,

$$\begin{aligned} \hat{H}_1^{-1} &= [1] \\ \hat{H}_2^{-1} &= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \\ \hat{H}_{2^{n+1}}^{-1} &= \begin{bmatrix} \hat{H}_{2^n}^{-1} & 0 \\ \hat{H}_{2^n}^{-1} & \hat{H}_{2^n}^{-1} \end{bmatrix} \end{aligned} \tag{40}$$

To summarize, we derived the following transformations and presented them in a recursive form,

1. From a boolean function to an equivalent polynomial over $\{1,-1\}$, and the inverse transformation (24).

2. From a boolean function to an equivalent polynomial over $\{0,1\}$ by (39), and the inverse transformation (40).

3. From a polynomial over $\{1,-1\}$ to a polynomial over $\{0,1\}$ by (34), and the inverse transformation (35).

The representation theory developed above can be used for representing error-correcting codes in a way that generalizes the representation that is described in Section 4. Consider the linear $[n,k]$ block code C. The code C can be represented by viewing each coordinate of the code as a boolean function of k variables. A vector $V \in C$ iff there exists a vector $X \in \{1,-1\}^k$ such that

$$V = (f_1(X), f_2(X), \ldots, f_n(X))$$

Clearly, the boolean functions associated with the coordinates of a linear block code are determined by the basis by which the code is represented. For linear block codes, every coordinate f_i corresponds to an XOR operation of some variables (according to the basis of the code). Thus, for every i, the boolean function f_i can be transformed by the method devised by Theorem 6.1 to an equivalent polynomial over $\{1,-1\}^k$ which consists of one monomial only. By the same argument as in Theorem 4.1, the MLD of a given word W is equivalent to solving the following maximization problem, with $X \in \{1,-1\}^k$,

$$\max\left(\sum_{i=1}^{n} W_i f_i(X)\right) \tag{41}$$

Observation 1: By Theorem 6.1, every monomial corresponds to a row in a Hadamard matrix. Since every f_i corresponds to a monomial it follows that every coordinate i of a linear block code corresponds to a row in a the Hadamard matrix of order 2^k. By definition, the first order Reed-Muller code (see Section 4) consists of all the possible 2^k monomials. Thus, the $[2^k,k]$ first order Reed-Muller code is the set of all rows of the Hadamard matrix of order 2^k. Hence, every linear block code is a punctured first order Reed-Muller code.

Observation 2: The MLD problem is equivalent to finding a codeword such that its inner product with the received word is maximal over all codewords (using the $\{1,-1\}$ representation). By Observation 1, the MLD of a first order Reed-Muller code is equivalent to finding an entry with maximal value in the vector $H_{2^k}W$.
For a general linear block code: a linear block code is a punctured first order Reed-Muller code (by Observation 1). Thus, we can construct a vector $\hat{W}$ of length 2^k (using W) such that it has zeros in all coordinates that do not correspond to a coordinate of the code. The MLD problem of W is again equivalent to finding an entry with maximal value in the vector $H_{2^k}\hat{W}$. Hence, the Fast Hadamard Transform [15] can be used efficiently to decode any low rate linear block code.

Observation 3: The following simple facts about an $[n, k]$ linear block code follow directly from Observation 1:

- The vectors of length 2^k which represent the coordinates of a linear block code are orthogonal since they correspond to columns of a Hadamard matrix.
- The number of 1's is equal to the number of -1's in each coordinate since the coordinates correspond to columns of a Hadamard matrix.
- The minimum weight (distance) of a first order Reed-Muller code is $2^{k-1} = 0.5n$ since the codewords are the rows of the Hadamard matrix of order 2^k.

Observation 4: The MLD problem as defined by (41) holds also for nonlinear codes. For nonlinear codes, a coordinate f_i can consist of more than one monomial. For example, consider the following nonlinear code of 4 codewords.

$$\mathcal{C} = [(00100), (11111), (10101), (01011)]$$

Then,

$$\begin{aligned} f_1 &= x_1x_2 \\ f_2 &= x_1 \\ f_3 &= 0.5(-1 - x_1 - x_2 + x_1x_2) \\ f_4 &= x_1 \\ f_5 &= 0.5(-1 + x_1 + x_2 + x_1x_2) \end{aligned}$$

From the above generalization, it follows that both for linear and nonlinear codes the MLD problem is equivalent to a maximization of a polynomial over $\{1, -1\}$. Hence, the following rather surprising theorem follows:

Theorem 6.3 *The following 3 problems are equivalent:*

1. *Maximization of polynomials with rational coefficients over the k-cube.*
2. *The MLD problem of an $[n, k]$ linear block code.*
3. *The MLD problem of a block code not necessarily linear that consists of 2^k codewords.*

7 Solving 0-1 nonlinear programming problems using decoding techniques

An unconstrained **nonlinear 0-1 program** [11] is a problem of the form:

$$\max \left(\sum_{i=1}^{n} w_i \prod_{j \in S_i} X_j \right) \tag{42}$$

where S_i is a subset of $\{1,\ldots,n\}$, and $X_j \in \{0,1\}$ for all i. Basically, the problem in (42) is a problem of finding a maximum of a 0-1 polynomial. A special case of (42) is the quadratic polynomial over $\{-1,1\}$ which was presented in Section 2. Clearly, every polynomial over $\{-1,1\}$ can be transformed to an equivalent one over $\{0,1\}$ by a change of variable as discussed in Section 6. The maximization of a quadratic polynomial over $\{0,1\}$ is known to be an NP-hard problem [6]. One of the ways to prove it is by showing that the Maximum Cut in a graph problem can be reduced to it. The reduction is based on the same technique which is used in Section 2 to show the equivalence between quadratic energy functions and cuts in graph.

The problem in (42) was studied extensively [9,11]. The main effort concentrated in identifying special cases which are solvable in polynomial time [12] and in devising approximation techniques [10].

The most common technique for solving unconstrained 0-1 programs is by transforming them to the problem of finding the maximum weight independent set in a graph [2,19]. Finding the maximum weight independent set in a graph is NP-hard, but there are some solvable cases. For example, the problem is solvable in polynomial time (by min cut-max flow techniques) if the graph is bipartite [2]. A known class of problems, like (42), which are solvable in polynomial time, are those problems which correspond to finding the maximum weight independent set in a bipartite graph.

Definition: Let $\hat{G} = (\hat{V}, \hat{E})$ be a graph, S is an *independent set* of nodes in the graph iff $S \subseteq \hat{V}$ and no two nodes of S are connected by an edge. Suppose that every node in $\hat{V}$ is assigned a positive integer called the weight of a node. The problem of finding an independent set of nodes such that the sum of its weights is maximal over all possible independent sets, is known as the *maximum weight independent set* problem.

The problem in (42) is transformed to the problem of finding the maximum weight independent set by using the concept of a *conflict graph* of a 0-1 polynomial [2,19]. The idea will be presented by the following example.

Example: Consider the following 0-1 polynomial,

$$f = -2X_1 - 2X_2 + 5X_1X_2 - 4X_1X_2X_3 \tag{43}$$

One can show that f can be transformed to an equivalent polynomial such that all the terms (except the constant one) have positive coefficients. The new polynomial involves both the variables and their complements. This is done by noticing that:

$$X = 1 - \bar{X}$$

Hence,

$$f = -4 + 2\bar{X}_1 + 2\bar{X}_2 + X_1X_2 + 4X_1X_2\bar{X}_3 \tag{44}$$

Clearly, maximization of f is equivalent to maximization of f without the constant term; so the constant term can be neglected.

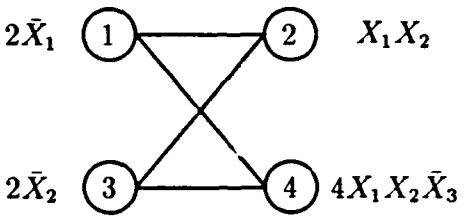

Figure 3: The conflict graph associated with f.

The *conflict graph*, to be denoted by $\hat{G}(f)$, associated with a polynomial f has a node set which corresponds to the terms of f, one node to a term (but the constant term). Two nodes in $\hat{G}(f)$ are connected by an edge iff one of the corresponding terms contains a variable and the other corresponding term contains the same variable complemented. The weight of a node in $\hat{G}(f)$ is the coefficient of the corresponding term in f. Figure 3 shows the conflict graph associated with f above.

The maximum weight independent set of $\hat{G}(f)$ is $\{2,4\}$; that is, the nodes that correspond to X_1X_2 and to $X_1X_2\bar{X}_3$. The weight of the set is 5, the assignment which achieves the maximum corresponds to $X_1 = 1$, $X_2 = 1$ and $X_3 = 0$. Thus, the maximum of f is -4+5=1.

One can prove that the above procedure works in the general case; that is:

1. Every maximum weight independent set in $\hat{G}(f)$ corresponds to a maximum in f (and vice versa), with the values of the terms associated with the nodes in the set equal to 1.

2. In general the problem of finding the maximum weight independent set in a graph is solvable in polynomial time for bipartite graphs (the graph in Figure 3 is bipartite).

3. The conflict graph associated with a polynomial is not unique, because a term can be made positive by complementing any odd number of its variables.

In the following we will show how decoding techniques can be used to maximize 0-1 nonlinear programs. Consider the 0-1 polynomials associated with Hamming codes (see Section 4). The family of these polynomials will be denoted by HP (*Hamming Polynomials*) . It will be shown, by an example, that HP is not contained in the family of polynomials related to bipartite conflict graphs. Thus, HP is not a subset of the family of polynomials whose maximization is known to be easy.

Consider the following polynomial over $\{0,1\}$:

$$M = 3-6X_1-2X_2-2X_3-6X_4+4(X_1X_2+X_1X_3+X_1X_4+X_2X_3+X_2X_4+X_3X_4)-8X_2X_3X_4 \quad (45)$$

The polynomial M is not associated with a bipartite conflict graph, as stated in the next proposition.

Proposition 7.1 *There does not exist $\hat{G}(M)$, a conflict graph associated with M, which is bipartite.*

Proof: The proof is straightforward, it follows from checking all the possible ways to convert the sign of the cubic term. □

A maximum of a polynomial which belongs to HP can be found by applying the decoding procedure for Hamming codes. Also, there is an efficient method to recognize if a given polynomial is in HP. We will describe both the recognition procedure and the maximization procedure by continuing with the above example.

Proposition 7.2 *The polynomial M is a Hamming polynomial.*

Proof:

1. Transform M to an equivalent polynomial over $\{-1, 1\}$ by a change of variable $U = 0.5(1 - X)$ (as in Section 6).

$$M = -U_2 - U_3 + U_4 + U_1U_2 + U_1U_3 + U_1U_4 + U_2U_3U_4 \tag{46}$$

2. By the derivation in Section 4, it is clear that M is equivalent to MLD of

$$\omega = (1,1,0,0,0,0,0)$$

with regard to the code defined by the following generator matrix:

$$G = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{47}$$

3. The matrix G can be brought to a systematic form, to be denoted by $\tilde{G}$, by row operations:

$$\tilde{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \tag{48}$$

From $\tilde{G}$ we obtain $\tilde{H}$, the systematic parity check matrix (see Section 5):

$$\tilde{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{49}$$

The polynomial M is a Hamming polynomial because its parity check matrix contains all the possible columns (but the all-0 column).

4. To decode ω we will use the syndrome, that is:

$$\omega \tilde{H}^T = (0,1,0)$$

The error is in the location which correspond to the row in $\tilde{H}^T$ that is equal to the syndrome. Hence, the result of the decoding is: $(1,1,0,0,0,1,0)$. The maximum is attained at $X^* = (1,1,1,0)$, and $M(X^*) = 5$.

By the above procedure we proved that M is in HP and found its maximum. □

A few remarks with regard to the above procedure:

1. The procedure in the above proof can be applied to a general 0-1 polynomial. Consider the polynomial representation over $\{-1,1\}$ (the one obtained after step 1 above), a necessary condition that a polynomial is in HP is that the absolute values of the coefficients in the $\{-1,1\}$ representation are equal (the constant is neglected).

2. The complexity of the recognition process is determined by the complexity of the transformation from the $\{0,1\}$ representation to the $\{1,-1\}$ (step 1). This transformation is exponential in the degree of the polynomial over $\{0,1\}$.

By Section 4, maximization of polynomials over $\{-1,1\}$ with coefficients in $\{-1,1\}$ is equivalent to MLD problem of linear block codes. The generalization to polynomials that have integer (or rational) coefficients follows immediately by expressing a term with a coefficient being equal to a (a positive integer) as a identical terms with coefficients equal to 1.

To summarize, we have established a technique for solving 0-1 nonlinear programs by decoding techniques. In particular, for the family of Hamming polynomials: it was proven that this family of polynomials is not a subset of the family of polynomials which are associated with bipartite conflict graphs, and both a recognition procedure and a solution procedure were derived.

References

[1] P. Baldi, PhD Thesis, Caltech 1986.

[2] M. L. Balinski, *On A Selection Problem*, Manag. Scien., Vol. 17, No. 3, pp. 230-231, Nov. 1970.

[3] N. Biggs, *Algebraic Graph Theory*, Cambridge University Press, 1975.

[4] J. Bruck and J. Sanz, *A Study on Neural Networks*, IBM ARC Computer Science, RJ 5403, 1986. To appear in International Journal of Intelligent Systems.

[5] J. Bruck and J. W. Goodman, *A Generalized Convergence Theorem for Neural Networks and its Applications in Combinatorial Optimization*, IEEE First ICNN, San Diego, June 1987. An extended version was submitted to the IEEE Transactions on Information Theory.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[7] E. Goles, F. Fogelman and D. Pellegrin, *Decreasing Energy Functions as a Tool for Studying Threshold Networks*, Disc. Appl. Math., 12, pp. 261-277, 1985.

[8] S. L. Hakimi and H. Frank, *Cut-Set Matrices and Linear Codes*, IEEE Transactions on Information Theory, Vol. IT-11, pp. 457-458, July 1965.

[9] P. L. Hammer and S. Rudeanu, *Boolean Methods in Operations Research*, Springer-Verlag, NY, 1968.

[10] P. L. Hammer, P. Hansen and B. Simeone, *Roof Duality, Complementation and Persistency in Quadratic 0-1 Optimization*, Math Prog. 28, pp. 121-155, 1984.

[11] P. Hansen, *Methods of Nonlinear 0-1 Programming*, Annals of Discrete Math. 5, pp. 53-70, 1979.

[12] P. Hansen and B. Simeone, *Unimodular Functions*, Disc. Appl. Math. 14, pp. 269-281, 1986.

[13] J. J. Hopfield, *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, Proc. Nat. Acad. Sci. . USA, Vol. 79, pp. 2554-2558, 1982.

[14] J. J. Hopfield and D. W. Tank, *Neural Computations of Decisions in Optimization Problems*, Biol. Cybern. 52, pp. 141-152, 1985.

[15] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, New York, 1977.

[16] A. Mukhopadhyay, *Recent Development in Switching Theory*, Academic Press, 1971.

[17] W. W. Peterson and W. J. Weldon, *Error-Correcting Codes*, The MIT Press, Cambridge, Mass., 1971.

[18] J. C. Picard and H. D. Ratliff, *Minimum Cuts and Related Problems*, Networks, Vol 5. pp. 357-370, 1974.

[19] J. M. W. Rhys, *A Selection Problem of Shared Fixed Costs and Network Flows*, Manag. Scien., Vol. 17, No. 3. pp. 200-207, Nov. 1970.

Harmonic Analysis of Polynomial Threshold Functions *

Jehoshua Bruck

Information Systems Laboratory

EE Department, Stanford University

Stanford, CA 94305

Abstract

A Boolean function is polynomial threshold if it can be represented as a sign function of a polynomial that consists of a polynomial (in the number of variables) number of terms. The main result is showing that the class of polynomial threshold functions (which we call PT_1) is strictly contained in the class of Boolean functions that can be computed by a depth 2, unbounded fan-in polynomial size circuit of linear threshold gates (which we call LT_2).

We use harmonic analysis of Boolean functions to derive a necessary and sufficient condition for a function to be an S-threshold function for a given set S of monomials. We use this condition to show that the number of different S-threshold functions, for a given S, is at most $2^{n|S|}$. These results turn out to be a generalization of known results for linear threshold functions.

Based on the necessary and sufficient condition we derive a lower bound on the number of monomials in a threshold function. The lower bound is expressed in terms of the spectral representation of a Boolean function. We find that Boolean functions that have an exponentially small spectrum are not polynomial threshold. We exhibit a family of functions that has an exponentially small spectrum; we call them 'semi-bent' functions. We construct a function that is both semi-bent and symmetric to prove that PT_1 is properly contained in LT_2.

We also extend the lower bound technique to depth 2 circuits of linear threshold gates.

*Submitted to SIAM Journal on Discrete Mathematics, 1988.

1 Introduction

A Boolean function $f(X)$ is a *threshold function* if

$$f(X) = sgn\,(F(X)) = \begin{cases} 1 & \text{if } F(X) > 0 \\ -1 & \text{if } F(X) < 0 \end{cases}$$

where

$$F(X) = \sum_{\alpha \in \{0,1\}^n} w_\alpha X^\alpha$$

and

$$X^\alpha \stackrel{\text{def}}{=} \prod_{i=1}^{n} x_i^{\alpha_i}$$

Throughout this paper a *Boolean function* will be defined as $f : \{1,-1\}^n \longrightarrow \{1,-1\}$; namely, 0 and 1 are represented by 1 and -1, respectively. It is also assumed, without loss of generality, that $F(X) \neq 0$ for all X.

A *threshold gate* is a gate that computes a threshold function. It can be shown that any Boolean function can be computed by a single threshold gate if we allow the number of monomials in $F(X)$ to be as large as 2^n. Although such a result is not interesting by itself, it stimulates the following natural question: What happens when the number of monomials (terms) in $F(X)$ is bounded by a polynomial in n?

The question can be formulated by defining a new complexity class of Boolean functions. This class, called PT_1 for *Polynomial Threshold* functions, is made of all the Boolean functions that can be computed by a single threshold gate in which the number of monomials is bounded by a polynomial in n. The main goal of this paper is the study of this complexity class and its relations with other known complexity classes of Boolean functions.

More precisely, let $S \subseteq \{0,1\}^n$; a Boolean function f is an *S-threshold* function if there exist integers that we call weights (the w_α's) such that $f(X) = sgn(\sum_{\alpha \in S} w_\alpha X^\alpha)$. Hence, a Boolean function $f(X)$ is in PT_1 if there exists a set S, with $|\,S\,|$ bounded by polynomial in n, such that $f(X)$ is S-threshold. Notice that there is no restriction on the size of the

weights.

A related class of functions is the class of *linear threshold* functions [6, 10]. A Boolean function is *linear threshold* if it is S-threshold with S corresponding to the constant and linear monomials. We define LT_1 to be the class of functions that are computable by a single linear threshold gate.

The next step is to define complexity classes which relate to circuits. Define LT_d (PT_d) to be the class of Boolean functions that can be computed by an unbounded fan-in polynomial size circuit of depth at most d which consists of linear (polynomial) threshold gates.

Recently, there has been a considerable interest in study of the computational model of bounded depth unbounded fan-in polynomial size circuits that consist of linear threshold gates [5, 11, 13]. This interest follows from recent results in complexity of circuits [7, 12, 15] which indicate that MAJORITY (hence, linear threshold functions) can not be computed by a bounded depth unbounded fan-in polynomial size circuit that consists of $\vee$, $\wedge$, NOT and PARITY gates. Thus, the next natural step in the analysis is adding MAJORITY as a possible gate in the computational model. Notice that in the results in [5] the weights are bounded by a polynomial in n. To make the distinction from the case in which the weights are not bounded we put 'hats'. Namely, $\hat{LT}_d$ and $\hat{PT}_d$ correspond to circuits with bounded weights. Using this notation, a related result in [5] is:

$$\hat{LT}_1 \subset \hat{LT}_2 \subset \hat{LT}_3$$

In this context, the study of circuits of polynomial threshold functions can be viewed as study of a model in which a single gate is rather powerful. Namely, there is no 'trivial' function that cannot be computed by a single gate. For example, PARITY, EXACT_k (output -1 iff k of its inputs are -1) and the characteristic function of a linear subspace (code) [1, 9] are all in PT_1 but none of them is in LT_1 (see Appendix A). This fact suggests that the separation between the classes PT_1 and PT_2 is not going to be an easy problem.

The main result in the paper is a characterization of the power of PT_1 with respect to the

hierarchy of circuits of linear threshold functions. We have:

$$LT_1 \subset PT_1 \subset LT_2$$

which also implies that $PT_1 \subset PT_2$.

Clearly, $LT_1 \subset PT_1$ follows from the fact that PARITY is not in LT_1. But, $PARITY(X) = sgn(x_1 x_2 \ldots x_n)$, hence it is in PT_1. Proving that $PT_1 \subseteq LT_2$ is based on the observation that PARITY does not require the full strength of a depth 2 circuit of linear threshold elements and is described in Section 2. In order to prove that this containment is proper we developed a technique for deriving lower bounds for the number of monomials in a threshold logic function. This technique is based on the spectral representation of a Boolean function. Most of the paper is devoted to the development of this technique and its applications.

In Section 3 we review the subject of harmonic analysis of Boolean functions [8] and show that every Boolean function has a representation as a polynomial over the rationals and hence as a threshold function.

In Section 4 we use the spectral representation of Boolean functions and derive a necessary and sufficient condition for a function to be an S-threshold function for a given S. We use this condition to show that the number of different S-threshold functions, for a given S, is at most $2^{n|S|}$. These results turn out to be a generalization of known results for linear threshold functions [2, 6, 10].

In Section 5 we use the necessary and sufficient condition to derive a lower bound on the number of monomials in a threshold function. The lower bound is expressed in terms of the spectral representation of a Boolean function. We find that Boolean functions that have an exponentially small spectrum are not polynomial threshold.

In Section 6 we exhibit a family of functions that has an exponentially small spectrum; we call them 'semi-bent' functions. We construct a function which is both semi-bent and symmetric to prove that PT_1 is properly contained in LT_2.

In Section 7 we show how the lower bound technique can be extended to get a result in [5]

that $\hat{LT}_2 \subset \hat{LT}_3$. Hence, for bounded weight circuits we have:

$$\hat{LT}_1 \subset \hat{PT}_1 \subset \hat{LT}_2 \subset \hat{PT}_2$$

Finally, we address some open problems.

2 Simulation of Polynomial Threshold Functions

It is a well known result that PARITY (as well as other symmetric functions) is in LT_2 [5, 11]. From this fact it follows that a polynomial threshold function can be simulated by a depth 3 circuit of linear threshold gates. The idea is to compute the monomials using depth 2 circuits and combine the monomials in the gate in the third layer.
What we will show here is that depth 2 is enough:

Theorem 2.1

$$PT_1 \subseteq LT_2$$

Proof: The idea is to notice that PARITY does not require the full power of a depth 2 circuit of linear threshold gates. Actually, PARITY can be realized by a set of linear threshold elements in the first layer while in the second layer we need only to sum and add a constant to get the desired result. Namely, we do not have to use the threshold operation in the second layer.

Example: Let $f(X) = x_1 \oplus x_2$. Let $F_1(X) = -1 - x_1 - x_2$ and $F_2(X) = -1 + x_1 + x_2$. It can be verified that:

$$f(X) = 1 + sgn(F_1(X)) + sgn(F_2(X))$$

Note that we are using the $\{1, -1\}$ representation instead of $\{0, 1\}$, respectively.
The above is true in general for PARITY of n variables. In the general case we need up to $n + 1$ linear threshold gates in the first layer and again only summation and addition of a constant in the second layer. Using this observation a polynomial threshold function can be

simulated by depth 2 linear threshold circuit in a way similar to that done with depth 3. □

Proving containment of polynomial threshold functions in LT_2 turns out to be a very easy problem compared to the problem of proving that this containment is proper; the latter requires proving lower bounds. The rest of the paper is devoted to the development of a technique for getting lower bounds for polynomial threshold functions and the application of this technique to getting separation results.

3 Polynomial Representation of Boolean Functions

In this section the representation of Boolean functions as polynomials over the field of rational numbers is presented.

Definition: A *Hadamard matrix* of order m, to be denoted by H_m, is an $m \times m$ matrix of +1's and -1's such that:

$$H_m H_m^T = m I_m \tag{1}$$

where I_m is the $m \times m$ identity matrix. The above definition is equivalent to saying that any two rows of H are orthogonal.

Hadamard matrices of order 2^k exist for all $k \geq 0$. The so called Sylvester construction is as follows [9]:

$$\begin{aligned} H_1 &= [1] \\ H_2 &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \\ H_{2^{n+1}} &= \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix} \end{aligned} \tag{2}$$

Definition: Given a Boolean function f of order n, p is a polynomial (with coefficients over the field of rational numbers) *equivalent* to f iff for all $X \in \{1, -1\}^n$:

$$f(X) = p(X)$$

As an example, let $f = x_1 \oplus x_2$; that is, f is the XOR function of two variables. It is easy to check that in the $\{1, -1\}$ representation $p(x_1, x_2) = x_1 x_2$.

Notice that for every Boolean function f, the polynomial p is linear in each of its variables because $x^2 = 1$ for $x \in \{-1, 1\}$. It is known that every Boolean function has a unique representation as a polynomial [1, 8] . This representation is derived by using the Hadamard matrix, as described by the following theorem.

Theorem 3.1 *Let f be a Boolean function of order n. Let p be a polynomial equivalent to f. Let A_{2^n} denote the vector of coefficients of p. Let P_{2^n} denote the vector of the 2^n values of p (and f). Then:*

1. *The polynomial p always exists and is unique.*
2. *The coefficients of p are computed as follows,*

$$A_{2^n} = \frac{1}{2^n} H_{2^n} P_{2^n}$$

Proof: The proof is constructive. The idea is to compute A_{2^n} by solving a system of linear equations. Let us start by computing the coefficients of p, for f being a function of one variable:

$$p(x_1) = a_0 + a_1 x_1$$

and,

$$\begin{aligned} p(1) &= a_0 + a_1 \\ p(-1) &= a_0 - a_1 \end{aligned}$$

Clearly,

$$P_2 = H_2 A_2 \tag{3}$$

and by (1),

$$A_2 = \frac{1}{2} H_2 P_2 \tag{4}$$

The above can be generalized to n variables by induction on n. Assume true for n

$$P_{2^n} = H_{2^n} A_{2^n}$$

For (n+1), consider the different values of x_{n+1} and get

$$\begin{aligned} P_{2^{n+1}} &= \begin{bmatrix} H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} \end{bmatrix} A_{2^{n+1}} \\ &= H_{2^{n+1}} A_{2^{n+1}} \end{aligned}$$

Hadamard matrices are nonsingular; thus, for any given f a unique p (defined by the vector of coefficients A_{2^n}) always exists. □

Example: Consider the function f of 3 variables,

$$f(x_1, x_2, x_3) = (x_1 \wedge x_3) \vee (x_1 \wedge x_2) \tag{5}$$

By Theorem 3.1,

$$p(x_1, x_2, x_3) = \frac{1}{8}(2 + 6x_1 + 2x_2 - 2x_1x_2 + 2x_3 - 2x_1x_3 + 2x_2x_3 - 2x_1x_2x_3) \tag{6}$$

The entries of the vector A are denoted by $\{a_\alpha \mid \alpha \in \{0,1\}^n\}$ and called the spectral representation of a function. Note that a_α is the coefficient of X^α in the polynomial representation where $X^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$.

The above method is applicable not only to Boolean functions but also to any function of the form $f : \{1, -1\}^n \longrightarrow \Re$.

4 Necessary and Sufficient Conditions

We use the polynomial representation of Boolean functions that was developed in the previous section to derive a necessary and sufficient condition for a function to be an S-threshold function, for arbitrary S. This result turned out to be a generalization of a known result for

linear threshold functions [2, 6, 10].

Let $f(X) = sgn(F(X))$ be a threshold function. Without loss of generality, assume that $F(X) \neq 0$ for all $X \in \{1,-1\}^n$. The following simple lemma enables us to express the relation between $f(X)$ and $F(X)$ in a global way. That is, instead of having 2^n conditions we have only one:

Lemma 4.1 *Let $f(X)$ be a Boolean function and let $F(X) \neq 0$ for all $X \in \{1,-1\}^n$, then*

$$f(X) = sgn(F(X)) \qquad \forall X \in \{1,-1\}^n \tag{7}$$

iff

$$\sum_{X \in \{1,-1\}^n} | F(X) | = \sum_{X \in \{1,-1\}^n} f(X)F(X) \tag{8}$$

Proof: Suppose there exists X_1 that violates (7); that implies ($F(X) \neq 0$ for all X) that

$$| F(X_1) | > f(X_1)F(X_1)$$

Hence, (8) is also not true because violation in equation (7) can only decrease the value on the right hand side of (8). Clearly, if (7) is true so is (8). □

Lemma 4.2

$$\sum_{X \in \{1,-1\}^n} X^\alpha = \begin{cases} 2^n & \text{if } \alpha = \textit{all-0 vector} \\ 0 & \textit{else} \end{cases} \tag{9}$$

Proof: Follows from the fact that X^α corresponds to a row of a Sylvester type Hadamard matrix (see Theorem 3.1). □

The necessary and sufficient condition follows from (8) by using the polynomial representation of a Boolean function.

Theorem 4.1 *Fix $S \subseteq \{0,1\}^n$. Let $F(X) = \sum_{\alpha \in S} w_\alpha X^\alpha$. Let $f(X)$, $X \in \{-1,1\}^n$, be a Boolean function with spectral representation $\{a_\alpha \mid \alpha \in \{0,1\}^n\}$. Then:*

$$f(X) = sgn(F(X)) \qquad \forall X \in \{1,-1\}^n$$

iff

$$\sum_{X\in\{1,-1\}^n} \mid F(X) \mid = 2^n \sum_{\alpha\in S} w_\alpha a_\alpha \tag{10}$$

Proof: By the Lemma 4.1 it is enough to show that

$$\sum_{X\in\{1,-1\}^n} f(X)F(X) = 2^n \sum_{\alpha\in S} w_\alpha a_\alpha$$

We write $f(X)$ as a polynomial:

$$f(X) = \sum_{\alpha\in\{0,1\}^n} a_\alpha X^\alpha$$

and get that the constant term of $f(X)F(X)$ is $\sum_{\alpha\in S} w_\alpha a_\alpha$. Hence, the result follows from Lemma 4.2. □

Theorem 4.1 is interesting because it suggests that an S-threshold function is fully characterized by the set of spectral coefficients that correspond to S.

Theorem 4.2 *Let f_1 and f_2 be S-threshold functions with $\{a^i_\alpha \mid \alpha \in \{0,1\}^n\}$, $i = 1, 2$, be their spectral representation, respectively. Then $f_1(X) = f_2(X)$ for all $X \in \{1,-1\}^n$ iff $a^1_\alpha = a^2_\alpha$ for all $\alpha \in S$.*

Proof: Suppose $f_1 = f_2$. By the uniqueness of the spectral representation (Theorem 3.1) we get the 'only if'. Now suppose $a^1_\alpha = a^2_\alpha$ for all $\alpha \in S$. By Theorem 4.1 and the assumption that both f_1 and f_2 are S-threshold we get that there exists a set of weights that satisfies (10) for both f_1 and f_2. Hence, $F_1(X) = F_2(X)$ for all $X \in \{1,-1\}^n$. □

Corollary 4.1 *Consider the set $S \subseteq \{0,1\}^n$. Let f_1 and f_2 be Boolean functions of n variables. If $a^1_\alpha = a^2_\alpha$ for all $\alpha \in S$, then either both f_1 and f_2 are S-threshold or both are not S-threshold.*

One application of the above is counting the number of different S-threshold functions.

Theorem 4.3 *For $S \subseteq \{0,1\}^n$. There are at most $2^{n|S|}$ different S-threshold functions.*

Proof: It can be shown that for all $\alpha \in \{0,1\}$, a_α can assume at most 2^n different values. Hence, there are at most $2^{n|S|}$ different sets of $|\ S\ |$ spectral coefficients. Thus, the result follows from Corollary 4.1 . □

The above turned out to be a generalization of a known [6, 10] upper bound on the number of linear threshold functions for which $|\ S\ | = n+1$.

5 Lower Bounds

The necessary and sufficient condition that is derived above is used to derive lower bounds on the number of monomials in a threshold function, again, by using the spectral representation. Let $f(X) = sgn(F(X))$ be an S-threshold function, namely,

$$F(X) = \sum_{\alpha \in S} w_\alpha X^\alpha$$

We want to find lower bounds for $|S|$ as a function of the spectral representation of $f(X)$.

Lemma 5.1 *For all $\alpha \in S$:*

$$2^n\ |\ w_\alpha\ | \leq \sum_{X \in \{1,-1\}^n} |\ F(X)\ |$$

Proof: First we prove the statement for α being the all-0 vector:

$$\begin{aligned}
\sum_{X \in \{1,-1\}^n} |\ F(X)\ | &= \sum_{F(X)>0} F(X) - \sum_{F(X)<0} F(X) \\
&= \sum_{X \in \{1,-1\}^n} F(X) - 2 \sum_{F(X)<0} F(X) \\
&\overset{(a)}{=} 2^n w_{00\ldots00} - 2 \sum_{F(X)<0} F(X) \\
&\geq 2^n w_{00\ldots00}
\end{aligned}$$

Note that (a) follows from Lemma 4.2. The proof for arbitrary α follows from the fact that $\mid X^{\alpha} \mid = 1$, hence:

$$\mid F(X) \mid = \mid X^{\alpha} \mid\mid F(X) \mid = \mid X^{\alpha} F(X) \mid$$

Hence, we can make any w_{α} be the constant term without changing the value of $\mid F(X) \mid$. If $w_{\alpha} < 0$ we take $-F(X)$ and get the result. □

Theorem 5.1 *Let $f(X) = sgn(\sum_{\alpha \in S} w_{\alpha} X^{\alpha})$ be an S-threshold function and let $\{a_{\alpha} \mid \alpha \in \{0,1\}^n\}$ be its spectral representation; then*

$$\begin{aligned} \mid S \mid &\overset{(i)}{\geq} (\sum_{\alpha \in S} p_{\alpha} \mid a_{\alpha} \mid)^{-1} \\ &\overset{(ii)}{\geq} \hat{a}^{-1} \end{aligned}$$

where

$$p_{\alpha} = \frac{\mid w_{\alpha} \mid}{\sum_{\alpha \in S} \mid w_{\alpha} \mid}$$

and

$$\hat{a} = \max_{\alpha \in S} \mid a_{\alpha} \mid$$

Proof: We first prove (i). By Theorem 4.1 and Lemma 5.1, for all $\alpha \in S$:

$$\mid w_{\alpha} \mid \leq \sum_{\alpha \in S} w_{\alpha} a_{\alpha}$$

We sum the above inequality over all $\alpha \in S$ and get:

$$\begin{aligned} \sum_{\alpha \in S} \mid w_{\alpha} \mid &\leq \mid S \mid \sum_{\alpha \in S} w_{\alpha} a_{\alpha} \\ &\leq \mid S \mid \sum_{\alpha \in S} \mid w_{\alpha} \mid\mid a_{\alpha} \mid \end{aligned}$$

So we get (i). For (ii), just notice that $p_{\alpha} \geq 0$ and $\sum p_{\alpha} = 1$. □

We summarize this section by:

Corollary 5.1 *Fix any $\epsilon > 0$. Let $f(X)$ be a Boolean function of n variables. If $\mid a_{\alpha} \mid \leq 2^{-\epsilon n}$ for all $\alpha \in \{0,1\}^n$, then, for n sufficiently large, $f(X)$ is not a polynomial threshold function.*

6 Separating by Semi-Bent Functions

We use Corollary 5.1 to get separation results by looking at functions that have an exponentially small spectrum.

Definition: A Boolean function $f(X)$ is called 'bent' [3, 9, 14] iff $\mid a_\alpha \mid = 2^{-\frac{n}{2}}$ for all $\alpha \in \{0,1\}^n$. Notice that bent functions are defined for even n only.

Proposition 6.1 *The Inner Product Mode 2 (IP2) function, i.e.,*

$$f(X) = (x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus \ldots \oplus (x_{2n-1} \wedge x_{2n})$$

is a bent function.

Proof: See [9]. A sketch of an alternative proof: it can be proven by induction on n that IP2 when written as a vector is actually an eigenvector (with eigenvalue = 2^n) of a Sylvester type Hadamard matrix of order 2^{2n}. Hence, $|a_\alpha| = 2^{-n}$ for all $\alpha \in \{0,1\}^n$. □

Theorem 6.1

$$PT_1 \subset PT_2$$

Proof: By Corollary 5.1 and Proposition 6 [illegible] PT_1. But it is in PT_2 the AND's are computed in the [illegible] level and the XOR is comput[illegible] the gate in the second layer. □

Definition: Let $\epsilon > 0$ [illegible] Boolean function $f(X)$ is called 'semi-b[illegible] $\alpha \in \{0,1\}^n$, $|a_\alpha|$ [illegible] Clearly, a bent function is also a semi-bent function.

Theorem 6.2

$$PT_1 \subset LT_2$$

Proof: The fact that $PT_1 \subseteq LT_2$ is proved in Theorem 2.1. To show that it is a proper containment we must find a function which is in LT_2 but not in PT_1. Every symmetric function is in LT_2 [5, 11] and every semi-bent function is not in PT_1 (Corollary 5.1). Hence, a natural candidate for such a function will be a symmetric semi-bent function. Indeed, a symmetric semi-bent exists for all n as stated in Proposition 6.2 below. □

Consider the function:

$$f(X) = (x_1 \wedge x_2) \oplus (x_1 \wedge x_3) \oplus \ldots \oplus (x_{n-1} \wedge x_n) \tag{11}$$

Hence, $f(X)$ consists of the sum mod 2 of all the $\binom{n}{2}$ possible AND's between pairs of variables. We call this function the Complete Quadratic (CQ) function. Clearly, CQ is a symmetric function.

Proposition 6.2 *CQ is bent for n even, and semi-bent for n odd.*

Proof: Actually we can compute the exact spectral representation of CQ for every n, see Appendix B.

7 Lower Bounds for Circuits

Here we use the necessary and sufficient condition to derive lower bounds for the number of gates in depth 2 circuits of linear threshold gates.

A depth 2 circuit consists of a single gate that we call t (its output is the output of the circuit) together with a set of k gates whose outputs are inputs to t. The i'th gate in the first level of the circuit is denoted by c_i. Hence, the function computed by the circuit is $t(X) = t(c_1(X), c_2(X), \ldots, c_k(X))$; where t and c_i for all $i \in \{1..k\}$ are linear threshold gates.

We use the same ideas as in Section 4 and get:

Theorem 7.1 *Let*

$$T(X) = w_0 + \sum_{i=1}^{k} w_i c_i(X)$$

and assume $T(X) \neq 0$ *for all* $X \in \{1,-1\}^n$. *Let* $t(X) = sgn(T(X))$. *The spectral representation of* $t(X)$ *is* $\{a_\alpha \mid \alpha \in \{0,1\}^n\}$ *and the spectral representation of* $sgn(T(X))$ *is* $\{b^i_\alpha \mid \alpha \in \{0,1\}^n\}$. *Then*

$$\sum_{X \in \{1,-1\}^n} |T(X)| = 2^n \left(w_0 a_{00\ldots00} + \sum_{i=1}^{k} w_i \sum_{\alpha \in \{0,1\}^n} a_\alpha b^i_\alpha \right) \quad (12)$$

Proof: The proof is similar to the proof of Theorem 4.1. □

We show that the result in [5] that $IP2 \notin L\hat{T}_2$ can be derived by using the fact that the left hand side of (12) is greater or equal 2^n, hence:

Proposition 7.1 *Let* $t(X) = t(c_1(X), c_2(X), \ldots, c_k(X))$, *then:*

$$k \geq \frac{1 - w_0 a_{00\ldots00}}{\hat{w}\hat{c}}$$

where

$$\hat{w} = \max_{i \in \{1\ldots k\}} |w_i|$$

and

$$\hat{c} = \max_{i \in \{1\ldots k\}} \left| \sum_{\alpha \in \{0,1\}^n} a_\alpha b^i_\alpha \right|$$

a_α *and* b_α *have the same meaning as in Theorem 7.1.*

Now let $t(X) = IP2$, the fact that $IP2 \notin L\hat{T}_2$ follows from:

1. $\hat{w}$ is bounded by a polynomial in n.
2. $a_{00\ldots00}$ for IP2 is exponentially small.
3. Using the lemma by Lindsey [4, p. 88] about Hadamard matrices it can be shown that $\hat{c}$ is exponentially small

The above technique is effective for getting lower bounds on k whenever $t(X)$ and the c_i's (a family of gates/circuits) result in $\hat{c}$ which is exponentially small.

8 Open Problems

The main open problem is to find the exact relation between the class $PT^0 = \bigcup_{d \in \mathcal{N}} PT_d$ and the well known class NC^1. Here, we were concentrating on the relation between subclasses of PT^0 and subclasses of LT^0, with the goal of getting separation in LT^0. In particular we have the following conjecture:

Conjecture: For all $d \in \mathcal{N}$: $LT_d \subset PT_d \subset LT_{2d}$

Acknowledgement: I would like to thank Professor J. W. Goodman for his helpful comments. The support of the U.S. Air Force Office of Scientific Research is gratefully acknowledged.

Appendices

A Examples

Here we give some examples of polynomial threshold functions which are not linear threshold functions. To show that a function is polynomial threshold we need to give the explicit $F(X)$, while proving that a function is not linear threshold requires the application of the necessary and sufficient condition developed in the paper. We will work with Boolean functions of n variables but the reader should think of n being arbitrary.

We start with a trivial example of the well known PARITY function.

Definition:

$$\text{PARITY}(X) = \begin{cases} -1 & \text{odd number of } -1\text{'s in } X \\ 1 & \text{otherwise} \end{cases}$$

Clearly, $\text{PARITY}(X) = x_1 x_2 \cdots x_n$, so we even do not need the threshold to compute PARITY. To show that PARITY is not in LT_1 just notice that the spectral coefficients which correspond to the constant and linear terms are all 0. So there is no $F(X)$, such that $F(X) \neq 0$ for all $X \in \{1, -1\}^n$. that will satisfy the necessary and sufficient condition, because the right hand side of equation (10) is 0.

The second example is a bit more complicated.

Definition: Define the following function over n variables,

$$\text{EXACT}_k^n(X) = \begin{cases} -1 & \text{if the number of } -1\text{'s in } X \text{ is } k \\ 1 & \text{otherwise} \end{cases}$$

Proposition A.1

$$EXACT_k^n \in PT_1$$

Proof: Consider the Boolean function of $2n$ variables that is defined as follows: It is -1 iff the number of -1's in X is equal to the number of 1's in X. Clearly, this function is

$\text{EXACT}_n^{2n}(X)$. We show that this function is in PT_1 and the proof follows by reducing EXACT_k^n to EXACT_n^{2n}.

Let

$$F(X) = (n-1) + x_1x_2 + x_1x_3 + \cdots + x_{2n-1}x_{2n}$$

Namely, $F(X)$ consists of a constant term (n-1) and all the $\binom{2n}{2}$ monomials of two variables. We will show that $\text{EXACT}_n^{2n}(X) = sgn(F(X))$.

Suppose X consists of $(n+m)$ -1's and $(n-m)$ 1's. Notice that we want $F(X) < 0$ iff $m = 0$. We calculate the value of $F(X)$ as a function of m. We look at the value of the terms (the constant term is excluded) and get that the number of terms in $F(X)$ that are -1 is exactly

$$n^2 - m^2 = (n+m)(n-m)$$

and the rest of the terms are 1. Hence, for X having m -1's,

$$\begin{aligned} F(X) &= (n-1) + \binom{2n}{2} - 2(n^2 - m^2) \\ &= 2m^2 - 1 \end{aligned}$$

Clearly, $\text{EXACT}_n^{2n}(X) = sgn(F(X))$.

Notice that $\text{EXACT}_k^n(X) = \text{EXACT}_n^{2n}(X, Y)$, where Y is a vector of length n that consists of k 1's and $(n-k)$ -1's, e.g.

$$Y = (\overbrace{11\ldots1}^{k}\overbrace{-1-1\ldots-1}^{(n-k)})$$

That is, $\text{EXACT}_k^n(X)$ is in PT_1. □

Now we show that EXACT_k^n is not linear threshold.

Proposition A.2

$$EXACT_k^n \notin LT_1$$

Proof: We will show that the function EXACT_n^{2n} is not in LT_1. It can be shown that the spectral coefficients of EXACT_n^{2n} that correspond to the linear terms are all 0 and the one

that corresponds to the constant term is

$$a_{00\ldots00} = \frac{2^{2n} - 2\binom{2n}{n}}{2^{2n}}$$

Assume the function EXACT_n^{2n} is in LT_1, use Theorem 4.1 and Lemma 5.1 and get that

$$2^n \mid w_{00\ldots00} \mid \leq 2^n \mid w_{00\ldots00} \mid\mid a_{00\ldots00} \mid$$

Hence, we get that $\mid a_{00\ldots00} \mid \geq 1$ which is a contradiction. □

The third example is related to linear codes [9].

Definition Let $\mathcal{C}$ be a linear $[n, k]$ block code. Then the characteristic function of $\mathcal{C}$ is

$$I_{\mathcal{C}}(X) = \begin{cases} -1 & \text{if } X \in \mathcal{C} \\ 1 & \text{otherwise} \end{cases}$$

Here we only state the result without a proof. The idea in the proof is to use the representation of linear block code that was developed in [1].

Proposition A.3 *Let $\mathcal{C}$ be a linear block code; then $I_{\mathcal{C}} \in PT_1$ and $I_{\mathcal{C}} \notin LT_1$.*

Example: See [1, 9]. Let C be the $[7, 4]$ Hamming code. The parity check matrix of $\mathcal{C}$ is

$$H^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$I_{\mathcal{C}}(X) = sgn(3 - x_1x_2x_4x_5 - x_1x_3x_4x_6 - x_2x_3x_4x_7)$$

In general, for ān $[n, k]$ code we need only $(n - k + 1)$ terms in $F(X)$ that are easily calculated from the parity check matrix of the code.

B The Spectrum of the Complete Quadratic Function

The Complete Quadratic function is defined in Section 6. Here we prove Proposition 6.2. We start by giving an equivalent definition of the function CQ,

Proposition B.1

$$CQ(X) = \begin{cases} 1 & \textit{no. of -1's in } X = 0 \textit{ or } 1 \textit{ mod } 4 \\ -1 & \textit{otherwise} \end{cases}$$

Proof: Suppose there are m -1's in X. Since a pair in equation (11) is -1 iff both variables are -1, we have exactly $\binom{m}{2}$ pairs which are -1. Hence, the value of CQ(X) is determined by the evenness of $\binom{m}{2}$ and the result follows. □

First we calculate the spectrum for the case when n is even.

Proposition B.2 *Let* $\{a_\alpha \mid \alpha \in \{0,1\}^n\}$ *be the spectral representation of* $CQ(X)$. *Assume that* n *is even, then*

$$\mid a_\alpha \mid = 2^{-\frac{n}{2}}, \qquad \forall \alpha \in \{0,1\}^n$$

Proof: The proof is by induction on n. For $n = 2$ we have

$$CQ(x_2, x_2) = \frac{1}{2}(1 + x_1 + x_2 - x_1 x_2)$$

Assume true for n and show that the statement is true for $(n+2)$. We use the same notation as in Section 3, namely, P_{2^n} represents the vector with the values of CQ and A_{2^n} represents the vector of the spectral coefficients of CQ. Using Proposition B.1 it can be shown that $P_{2^{n+2}}$ can be expressed as a function of P_{2^n}:

$$P_{2^{n+2}} = \begin{bmatrix} P_{2^n} \\ \hat{P}_{2^n} \\ \hat{P}_{2^n} \\ -P_{2^n} \end{bmatrix}$$

where

$$\hat{P}_{2^n} = \hat{X}_{2^n} \circ P_{2^n}$$

$\hat{X}_{2^n}$ is the vector representaion of PARITY$(X) = x_1 x_2 \cdots x_n$ and 'o' is bitwise multiplication. Hence, by Theorem 3.1

$$\begin{aligned} A_{2^{n+2}} &= \frac{1}{2^{n+2}} H_{2^{n+2}} P_{2^{n+2}} \\ &= \frac{1}{2^{n+2}} \begin{bmatrix} H_{2^n} & H_{2^n} & H_{2^n} & H_{2^n} \\ H_{2^n} & -H_{2^n} & H_{2^n} & -H_{2^n} \\ H_{2^n} & H_{2^n} & -H_{2^n} & -H_{2^n} \\ H_{2^n} & -H_{2^n} & -H_{2^n} & H_{2^n} \end{bmatrix} \begin{bmatrix} P_{2^n} \\ \hat{P}_{2^n} \\ \hat{P}_{2^n} \\ -P_{2^n} \end{bmatrix} \\ &= \frac{1}{2} \begin{bmatrix} \hat{A}_{2^n} \\ A_{2^n} \\ A_{2^n} \\ -\hat{A}_{2^n} \end{bmatrix} \end{aligned}$$

where $\hat{A}_{2^n}$ is the reflection of A_{2^n}. Hence, if the result is true for n it is also true for $n+2$.

□

Example: Using the above recursive description of the spectrum of $CQ(X)$ we can calculate A_{16} from A_4:

$$A_4^T = \frac{1}{2}(1, 1, 1, -1)$$

And

$$A_{16}^T = \frac{1}{4}(-1, 1, 1, 1, \quad 1, 1, 1, -1, \quad 1, 1, 1, -1, \quad 1, -1, -1, -1)$$

The above is true for n even. For n odd we have,

Proposition B.3 *Let* $\{a_\alpha \mid \alpha \in \{0,1\}^n\}$ *be the spectral representation of* $CQ(X)$. *Assume that* n *is odd then*

$$\mid a_\alpha \mid = 0 \quad or \quad 2^{-\frac{n+1}{2}} \qquad \forall \alpha \in \{0,1\}^n$$

The proof is similar to the even case. We use induction on n and can write the recursive description of the spectrum.

Example: Let $n = 3$ then

$$CQ(x_1, x_2, x_3) = \frac{1}{2}(x_1 + x_2 + x_3 - x_1x_2x_3).$$

References

[1] J. Bruck and M. Blaum, *Neural Networks, Error-Correcting Codes and Polynomials over the Binary n-Cube*, IBM Research Report, RJ 6003, 1987.

[2] C. K. Chow, *On the Characterization of Threshold Functions*, Proc. Symp. on Switching Circuit Theory and Logical Design, pp. 34-38, 1961.

[3] J. F. Dillon, *Elementary Hadamard Difference Sets*, in: Proc. 6th S-E conf. Combinatorics, Graph Theory and Computing, pp. 237-249, 1975.

[4] P. Erdös and J. Spenser, *Probabilistic Methods in Combinatorics*, Academic Press, 1974.

[5] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy and G. Turán, *Threshold Circuits of Bounded Depth*, 28th FOCS, pp. 99-110, 1987.

[6] S. Hu, *Threshold Logic*, University of California Press, 1965.

[7] D. S. Johnson, *The NP-Completeness Column: An Ongoing Guide*, J. of Algorithms, 7, pp. 289-305, 1986.

[8] R. J. Lechner, *Harmonic Analysis of Switching Functions*, in: A. Mukhopadhyay (ed.), *Recent Development in Switching Theory*, Academic Press, 1971.

[9] F. J. Macwilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, New York, 1977.

[10] S. Muroga, *Threshold Logic and its Application*, John Wiley & Sons Inc., 1971.

[11] I. Parberry and G. Schnitger, *Parallel Computation with Threshold Functions*, in: Structure in Complexity Theory, LNCS vol. 223, pp. 272-290, 1986.

[12] A. A. Razborov, *Lower Bounds for the Size of Circuits of Bounded Depth with Basis* $\{\wedge, \oplus\}$, Math. notes of the Acad. of Science of the USSR, 41:4, pp. 333-338, Sept. 1987.

[13] J. H. Reif, *On Threshold Circuits and Polynomial Computation*, Proc. 2nd Structure in Complexity Theory Conf., pp. 118-123, 1987.

[14] O. S. Rothaus, *On 'Bent' Functions*, J. Comb. Theory, 20A, pp. 300-305, 1976.

[15] R. Smolensky, *Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity*, 19th STOC, pp. 77-82, 1987.